GLUE GENERATION FRAMEWORK IN UNIFRAME FOR THE CORBA-JAVA/RMI INTEROPERABILITY

TR-CIS-0302-03

Kalpana Tummala May 25, 2004

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **MAY 2004**	2. REPORT TYPE	3. DATES COVERED **00-00-2004 to 00-00-2004**
4. TITLE AND SUBTITLE **Glue Generation Framework in Uniframe for the Corba-Java/Rmi Interoperability**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Indiana University/Purdue University,Department of Computer and Information Sciences,Indianapolis,IN,46202**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**170**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

GLUE GENERATION FRAMEWORK IN UNIFRAME FOR THE CORBA-JAVA/RMI INTEROPERABILITY

A Technical Report

Report Number

TR-CIS-0302-03

Submitted to the Faculty

Of Purdue University

By

Kalpana Tummala

In Partial Fulfillment of the

Requirements for the Degree

Of

Master of Science

May 2004

To Mom, Dad, and Dev.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank the many people who have helped make this project possible.

I would like to thank Professor Rajeev R. Raje, my advisor, for his encouragement and guidance through the course of my project. His immense knowledge and insights provided a strong foundation for this project.

My special thanks to Professor Andrew Olson for being on my advisory committee and providing me guidance during critical periods of my project. His painstaking efforts to review my report and work are greatly appreciated.

I would like to thank Professor Jiang Zheng for being on my advisory committee and providing valuable inputs towards the project.

I would like to thank all my colleagues for their assistance and cooperation during my project.

I would like to thank the U.S. Department of Defense and the U.S. Office of Naval Research for supporting this research under the award number N00014-01-1-0746. Many thanks to the faculty, staff and all my colleagues at the Department of Computer and Information Science for their cooperation and goodwill.

I would like to thank my family for their support. Last but not the least; I would like to thank my husband Dev for his encouragement and support throughout my stay at IUPUI.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Tummala, Kalpana, M.S., Purdue University, May, 2004. “Glue Generation Framework in UniFrame for the CORBA-Java/RMI Interoperability”. Major Professor: Rajeev Raje

Software realization of a Distributed Computing System (DCS) is typically achieved by integrating independently created and deployed components with one another to form a coalition of distributed software components. The UniFrame approach provides a seamless framework for achieving a DCS by automatic or semi-automatic integration of heterogeneous distributed components while taking into account their QoS (Quality of Service). Any integration of heterogeneous components raises interesting challenges, such as tackling differences in Syntax/Signature, Semantic, Protocol and the underlying component model. This project addresses the heterogeneity related to the underlying component models employed by Java RMI and CORBA. It proposes a framework to generate Glue code for interoperating between Java RMI and CORBA components. The Glue Generation architecture uses pre-defined code generation templates. The research also experiments with various choices for the placement of the generated glue code. A prototype is designed and experimented with to validate the Glue Generation architecture. The results obtained indicate that the Glue Generation architecture is comprehensive enough to enable the Java RMI and CORBA interoperability.

1. INTRODUCTION

Component-based software design has been a growing trend in the development of software solutions for distributed systems. A Distributed Computing System (DCS) is typically achieved by integrating independently created and deployed components, consisting of public interfaces and private implementations, with one another to form a coalition of distributed software components. Assembling such systems can benefit from either an automatic or semi-automatic integration of software components, taking into account the QoS (Quality of Service) constraints advertised by each component and the collection of components. The UniFrame Approach (UA) [2, 3] provides a framework that allows an interoperation of heterogeneous and distributed software components and incorporates the following key concepts: a) a meta-component model (the Unified Meta Model - UMM [1]), with a associated hierarchical setup for indicating the contracts and constraints of the components and associated queries for integrating a distributed system, b) an integration of the QoS at the individual component and distributed application levels, c) the validation and assurance of the QoS, based on the concept of event grammars, and e) generative rules, along with their formal specifications, for assembling an ensemble of components out of available component choices.

1.1 Heterogeneity

The UniFrame approach depends on the discovery and interoperation of independently deployed software components in a networked environment. These components, which may be heterogeneous, interact with each other to form a system to accomplish a task.

Heterogeneity problems can be divided into following:

❖ Syntax/Signature: difference in the representation of interface
❖ Semantic: difference in the meanings of the provided/required services of the components
❖ Protocol: ordering and blocking constraints that determine the availability of

services provided by components
- QoS Assurance

Heterogeneity also arises due to difference in technologies of the components i.e., difference in implementation language or object model. Due to the heterogeneity between different components, an architecture that provides the infrastructure to support the dynamic integration/interoperation is required. This project addresses the heterogeneity between components due to technology or object model and, in particular, addresses Java^{TM} Remote Method Invocation (RMI) and Common Object Request Broker Architecture (CORBA) technologies using a template-based approach.

In this project, an architecture and implementation for the interoperation aspect called the Glue Generation Framework in UniFrame for CORBA-Java/RMI Interoperability (UniGGen) is described. The UniGGen architecture semi-automates the process of generation of software code required for interoperation between selected technologically heterogeneous components meeting the necessary QoS requirements specified by a component assembler or system integrator. By using the UniGGen, the UniFrame Approach and its associated tools can achieve a semi-automatic construction of a distributed system.

1.2 Problem Definition and Motivation

The problem that needs to be solved is to provide for a Glue generation framework that is dynamic and encompasses services developed in diverse distributed computing models.

The motivation for creating a Glue generation framework is as follows: The software systems in any organization constantly undergo changes and evolutions. Moreover, these organizations may be geographically (or logically) dispersed, necessitating communication between independently created and deployed components, loosely integrating with one another to form a coalition of distributed software components. In order to deal with the constant evolutions and changes in software systems, there is a need to rapidly create software solutions for distributed environments using a component-based software development approach. The solution of decreeing a Components Off The Shelf (COTS) environment, in an organization, will require creating an ensemble of heterogeneous components, each adhering to some model. Assembling such systems can benefit from either an automatic or semiautomatic integration of software components.

The key to automating the process of assembling DCS is to have an infrastructure that allows for a seamless integration of different component models and sustains cooperation among heterogeneous components. Such an infrastructure will need to provide services to dynamically generate the code that will be required for communication between components of different models, meeting the necessary functional as well as non-functional requirements (such as desired levels of Quality of Service). The UniGGen architecture, proposed in this project, is designed to provide this infrastructure to support dynamic integration/interoperation.

One possible approach to achieve comprehensive interoperability is that of using a metamodel for heterogeneous distributed components. Web Services [7] are viewed as a possible solution to this problem. [8] describes Web Services as "Web Services are a standards-based software technology that lets programmers and integrators combine existing and new systems or applications in new ways over the Internet, within a company's boundaries, or across many companies. Web Services allow interoperability between software written in different programming languages, developed by different vendors, or running on different Operating Systems or platforms". Another possible approach to achieve interoperability is using already existing bridges commercially available. The UniGGen architecture is an independent approach using templates consisting of the glue code as an RMI-IIOP program and not the above two discussed approaches, since a protocol named RMI-IIOP has been already defined for interoperating between Java RMI and CORBA components hence that became the obvious choice. For the particular case of interoperating between Java RMI and CORBA components, the glue code generated is implemented in RMI-IIOP (RMI-Internet Inter-Orb Protocol) because of the following advantages:

- ❖ RMI-IIOP combines RMI-style ease of use with CORBA cross-language interoperability.
- ❖ RMI over IIOP includes the full functionality of a CORBA Object Request Broker (ORB). RMI-IIOP combines the best features of Java RMI with the best features of CORBA. It provides you with a powerful environment in which to do distributed programming using the Java platform.
- ❖ RMI-IIOP provides flexibility by allowing developers to pass any serializable Java object, i.e., provides flexibility to pass objects by value and also by reference between application components.
- ❖ IIOP eases legacy application and platform integration by allowing application components written in C++, Smalltalk, and other CORBA supported languages to communicate with components running on the Java platform.

1.3 Objectives

The specific objectives of this project are:

- ❖ Propose a generic architecture for interoperation between heterogeneous components.
- ❖ Develop a prototype to validate the proposed architecture/approach using components of Java RMI & CORBA technologies.
- ❖ Propose choices for the placement of the generated glue code.
- ❖ Study the overhead/effects of placement of the generated glue code on the performance of the system.

1.4 Contributions

The contributions of this project are:

- Provision of framework for semi-automatically generating the glue code required to interoperate between heterogeneous components implemented using Java RMI and CORBA object models using a template-based approach.
- Implementation of fully functional prototype for the UniGGen architecture.
- Study of the effect of the placement of the generated glue on the performance of the System composed.
- The architecture proposed achieves the interoperability between CORBA and Java RMI. Hence, templates can be used to solve the interoperability problem between components implemented in different underlying models.
- The process of designing templates requires manual intervention; however templates allow a more flexible approach to the generation of glue code with a higher degree of automation than is possible with a corresponding hand-crafted approach.
- The project suggests that the performance of the system composed depends on the placement of the glue and that the centralized placement causes bottleneck. The placement of glue with the respective initiator or responder components decreases the amount of overhead on the performance of the system composed.

1.5 Organization of this report

This project is organized into 6 chapters. The topic introduction consisting of the motivation, objectives, and the contributions of the project is presented in this chapter. Chapter 2 surveys background and related work. Chapter 3 provides an overview and design details of the project. Chapter 4 provides the implementation details of the project. Chapter 5 describes the prototype's validation by experimentation. Chapter 6 concludes this project with a discussion of what was accomplished and future work.

2. BACKGROUND AND RELATED WORK

In the previous chapter a brief introduction is presented, along with the problem definition, objectives and contributions of this project. This chapter provides an overview of the background and the work related to this project.

2.1 UniFrame

The UniFrame Approach (UA) strives to provide a flexible and effective framework for developing and implementing distributed computing systems. The framework unifies distributed component models under the Unified Meta-Component Model (UMM) [18]. Within the UMM exists heterogeneous components, service and quality of service guarantees, and the infrastructure. The components within UMM are autonomous and their implementation is nonuniform in the sense that there is no unified implementation framework, although they adhere to a distributed-component model. A UMM component consists of a state, an identity, a behavior, interfaces, private implementation, and three aspects which are computational, cooperative, and auxiliary. The computational aspect of a component is indicative of the task(s) completed, and depends on the task(s) objective, techniques utilized to achieve the task(s) objectives, and specification of the component's functionality. The cooperative aspect deals with the interaction between components and contains expected collaborators (components that can potentially cooperate), preprocessing collaborators (component is dependent on other components), and postprocessing collaborators (other components are dependent on this component). The auxiliary aspect handles additional features of a DCS, such as mobility, security, and fault tolerance.

Within UMM every component must specify the Quality of Service (QoS) specified for execution. The guarantee of QoS for each component is integral to any framework dependent on components being able to successfully complete the required tasks in a constantly changing heterogeneous environment. Within the UA, QoS of a DCS consists of a parameter catalog subdivided between static or design and dynamic or run-time parameters, formal specification of the parameters, and mechanism for ensuring the parameters [19].

The UniFrame Resource Discovery Service (URDS) provides the necessary infrastructure for UMM's discovery and communication mechanisms. Figure 2.1 displays the architecture of the URDS.

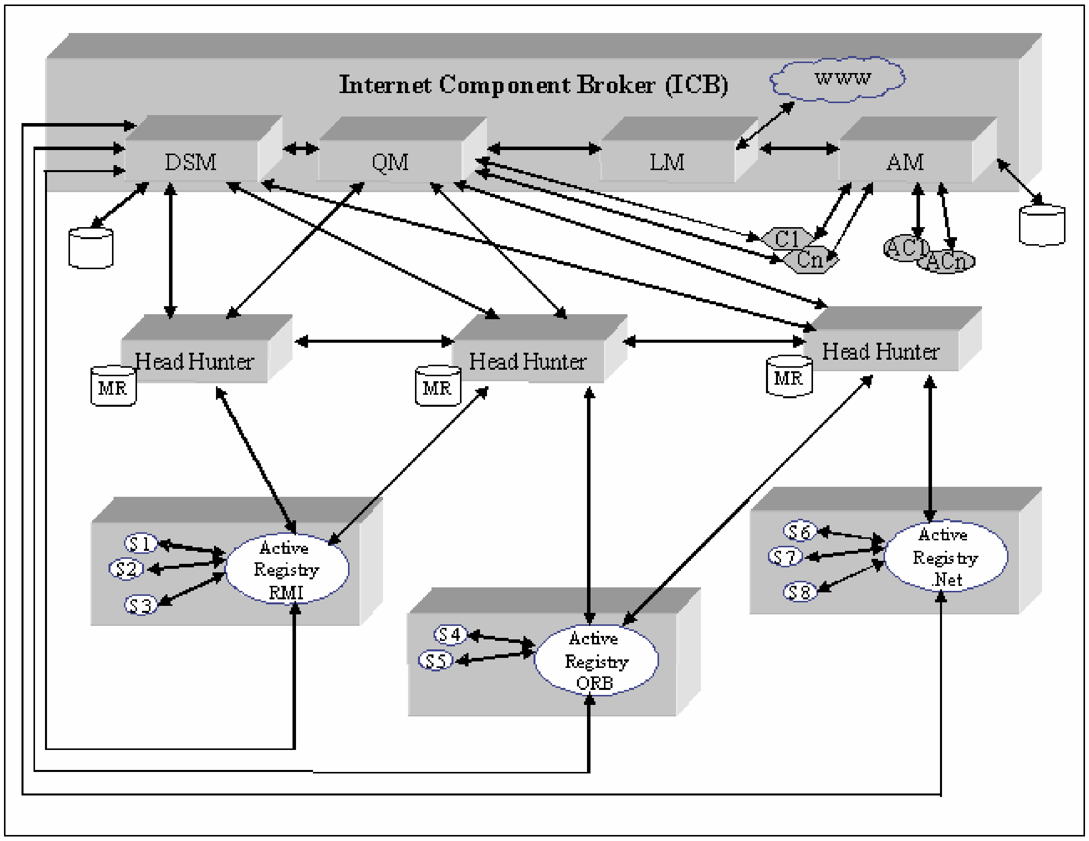


Figure 2.1 URDS Architecture (from [19])

The URDS consists of the Internet Component Broker (ICB) which in turn consists of the Query Manager (QM), Domain Security Manager (DSM), Link Manager (LM), and Adapter Manager (AM); Headhunters (HH); Meta-Repositories (MR); Active-Registries; Services; and Adapter Components. The federated hierarchy of the URDS architecture promotes scalability and fault tolerance. Within the URDS every ICB is broken down into sub-components in the hierarchy represented by the Headhunters and each ICB is linked together with unidirectional links. Discovery in URDS is scoped by an administratively defined domain whereas each domain refers to an industry sector (i.e. Health Care, Manufacturing, etc.), and each domain is supported by the sector or organization providing the URDS service. Finally, the URDS discovery process is based on periodic multicast announcements and access control to multicast address resources and data encryption are utilized for data transmission security. More extensive security features for URDS data communication are planned for the future.

2.2 GenVoca

GenVoca [5, 6] is formed by merging the designs of two independently-conceived software system generators: Genesis and Avoca. Genesis is the first building blocks technology for database management systems. Using a graphical layout editor, a customized DBMS can be specified by composing prefabricated software components. Avoca is a system for constructing efficient and modular network software suites using a combination of pre-existing and newly created communication protocols.

GenVoca is a domain-independent model for defining families of hierarchical systems as compositions of reusable components. GenVoca composes objects out of a series of layers, each of which handles a specific aspect of the object. These layers can be mixed and matched in a flexible way.

GenVoca is based on features such as components, realms and type equations. A component is a suite of interrelated variables, functions, and classes that work together as a unit to implement a particular feature of a software system for a given problem domain. A realm is a library of plug-compatible and interchangeable components, which is defined by a standardized interface consisting of functions and classes. Each interface defined represents a subsystem abstraction whose implementations are specified by families of subsystems (type equations), called an application family. All components of a realm inherit this interface and may specialize it by adding data and function members to existing classes, and by adding new variables, functions, and classes. The set of all component compositions defines a family of systems. Adding a new component to a realm is equivalent to adding a new rule to a grammar. Because large families of systems can be built using few components, GenVoca is a scalable model of software construction.

The UniGGen is designed for a distributed component system but GenVoca is suitable to model and create standalone programs. The UniGGen integrates heterogeneous components which GenVoca does not.

2.3 Babel

Babel developed by Center for Applied Scientific Computing (CASC) [9] addresses language interoperability problems using Interface Definition Language (IDL) techniques. An IDL describes the calling interface (but not the implementation) of a particular software library. Babel uses this interface description to generate glue code that allows a software library implemented in one supported language to be called from any other supported language. A Scientific Interface Definition Language (SIDL) has been designed that addresses the unique needs of parallel scientific computing. SIDL supports complex numbers and dynamic multi-dimensional arrays as well as parallel communication directives that are required for parallel distributed components. SIDL also provides other common features that are generally useful for software engineering, such as enumerated types, symbol versioning, name space management, and an object-oriented inheritance model similar to Java.

The Babel tool is divided into SIDL parser, code generator, small run-time support library, and the Alexandria [10] component repository. Alexandria is a research effort in the development, cataloging, and distribution of component-based software. It is used as the component repository in the Common Component Architecture infrastructure.

The Babel parser, which is available either at the command-line or through the Alexandria web interface, reads SIDL interface specifications and generates an intermediate XML representation.

The Babel code generator reads SIDL XML descriptions and automatically generates glue code for the specified software library. This glue code mediates differences among calling languages and supports efficient inter-language calls within the same memory address space and, eventually, across memory spaces for distributed objects. Four different types of files are created by code generator: stubs, skeletons, Babel internal representation, and implementation prototypes. The internal object representation is essentially a table of function pointers, one for each method in an object's interface, along with other information such as internal object state data, parent classes and interfaces, and Babel data structures. The code generators also create implementation files that contain function prototypes to be filled in by the library developers. Babel's run-time library provides reference counting and dynamic type identification. Babel supports Fortran 77, Fortran 90, C, C++, Python, and Java (client-side only).

Component applications will require support for parallel data redistribution to communicate object data between components running on differing numbers of processors. Babel does not support this parallel data redistribution.

2.4 SAGE

Systems and Applications Genesis Environment (SAGE) [11], released by Honeywell Inc. allows an engineer to develop applications efficiently on the High Performance Computing platforms.

SAGE provides a seamless programming interface between the software and the heterogeneous HPC platforms. SAGE provides a graphical and interactive interface for the creation of executable systems and applications based on customer defined specifications. SAGE allows an engineer to rapidly develop an application on a target system by trading and optimizing, modeling, and auto-generating code that ties the application to the target hardware. SAGE uses custom functional libraries to generate source code, which can be compiled and executed.

The SAGE glue-code generator is implemented in Alter, a programming language similar to Lisp in its syntax and style, which provides a direct interface to the contents of a SAGE model. The SAGE glue-code generator gains an access into the internal SAGE design tool environment, traverses objects in the models to filter relevant information, and then outputs the information in formats particular to the SAGE run-time source files. SAGE auto glue code generation and run-time components delivered and executed at 77.5 % of hand code versions. The designer needs to re-generate the glue code for other SAGE supported hardware platforms such as Mercury, CSPI, SIGI and SKY platforms. The UniGGen, at present does not support these platforms.

2.5 Jostraca

Jostraca [12] is a general purpose code generation toolkit for software developers using the Java Server Pages syntax. It uses the concepts of templates and domain models. Jostraca comes preconfigured for Java, Perl, Python, Ruby, and Rebol.

The actual code generation is a two-phase process. First, in the template compilation phase, the template is compiled into the valid source code of the template scripting language, into a program known as the CodeWriter. Second, in the code generation phase, the CodeWriter is then itself compiled and executed to output the generated source code of the target language. It also allows a clean and a complete separation of the template compilation from the code generation. So, this simple design allows Jostraca to use any external programming language as the template scripting language.

Jostraca attempts to solve the problems of the implementation domain. In order to serve many domain models and languages, Jostraca maintains programming language independence.

Jostraca's approach to generate software code is similar to that of the UniGGen. Jostraca does not provide a more abstract representation of templates themselves, so that templates can become structural units and can themselves be composed whereas the UniGGen generates the glue code using templates and the generated code is a first-class entity i.e., it can be composed to become part of the system.

2.6 Connector model

A Connector model is presented in [13], which provides a generic platform independent approach to automatic generation of connectors. They define a connector to be "an architectural element, which embodies communication among components in a system".

The Connector model provides a hierarchical model of connectors consisting of connector frame and connector architecture. A Connector frame is a topmost abstraction which provides the type of interconnections between components and connectors required for communication between the components to form a system. Connector architecture allows for describing the internal description or details of a connector i.e., the functionality of a connector in the form of composition of building blocks of connector implementation known as primitive elements. A primitive element provides a specific functionality.

A Connector generator which is a part of the connector architecture provides a simple API for communication, along with a deployment tool using which a developer can assemble an application. The Connector generator automatically generates connector code using the details provided through this deployment tool by selecting or choosing properties such as connected interfaces, parameters, technology, etc. All the intermediate processing from the definition of connector implementation to creating/generating code and deploying the connector is done using CDL, Component Definition Language which is similar to IDL, Interface Definition Language. The UniGGen architecture uses XML format to retrieve the specifications or the properties of the components required to generate the glue or connector code and finally deploys the glue code.

2.7 Piccola

In [14, 15] a small compositional or scripting language based on the Pi Calculus called Piccola is presented. It is used to compose applications from individual components. Its syntax is small and has a minimal set of features needed to specify different styles of software composition.

The main features of Piccola are communicating agents, which perform computations, and forms, which are values that are communicated as nested records. The communication agent's behavior is specified by scripts. Agents invoke services and compose forms. Forms and agents allow Piccola to unify components, static and dynamic contexts and arguments for invoking services whereas UniGGen uses templates to do so.

2.8 Bridges

This section provides a brief overview of a few commercial bridges that address the interoperability issue in the context of Java RMI and CORBA (in Java) technologies and the differences between the UniGGen architecture and these bridges.

2.8.1 JCorbaBridge

JCorbaBridge is created by OMEX (Object Middleware Experts) AG [16]. JCorbaBridge can be customized to bring already existing CORBA component services to J2EE based e-commerce platforms. It consists of:

- ❖ Core classes that provide the required framework.
- ❖ Generator that generates bridge components from CORBA-IDL files. The generated bridge component encapsulates the CORBA object and hides the complexity of CORBA programming from the Java developer.

JCorbaBridge/RMI provides the necessary infrastructure to bridge the gap between CORBA and Java systems. It also provides the ability to use a J2EE compliant application server as a runtime environment. It automatically generates stateless session beans for CORBA objects to be used within the JCorbaBridge. It has a few restrictions:

- It does not support the CORBA Any type.
- It does not support the CORBA objects as operation parameters.
- It does not support CORBA objects by value.

JCorbaBridge is not openly available software. Hence, the above analysis is based on information obtained on web.

2.8.2 Web services as an interoperability medium

Web Services are services and components that can be used on the Internet. Web services use XML-related technologies such as SOAP as the communication protocol, WSDL as the interface description language, and UDDI for registering and searching services. Data is marshaled into XML request and response documents and transferred between software packages using HTTP or message-based protocols. Web services technology can be used to interoperate between software applications i.e., transform the components to be composed as a system into web services and register them in UDDI. The components now can search each other in UDDI and can communicate with each other.

2.8.2.1 Capeclear web services software

CapeConnect [17], developed by Capeclear offers business integration solutions on a Web Services platform that exposes existing Java, Enterprise JavaBeans (EJB), and CORBA server-side components as Web Services i.e., CapeConnect composes a system of components using web services as an interoperability medium. CapeConnect accepts input from a client in the form of a SOAP message and converts it to Java or CORBA calls depending on the server-side component's underlying object model (Java, EJB or

CORBA). CapeConnect then retrieves the information from the server-side component that the client requires and transfers or returns it to the client as a SOAP response.

Thus, CapeConnect can only make Enterprise Beans, Java objects, and CORBA objects available to SOAP/XML clients i.e., CapeConnect enables communication between an XML or a SOAP direct client and a Java, EJB or CORBA server only. For two heterogeneous components, say Java client and CORBA server, the CapeConnect architecture has to be enhanced to enable communication between them i.e., the call or message sent from the client to the server has to be converted internally into an XML client.

2.9 Product Line Architecture for Component Model Domains

In [24, 25], as a part of UniFrame project, a formalism to automate middleware generation using Two-Level Grammar (TLG) is discussed.

The Internet Component Broker (ICB) [Raj01] organizations develop and maintain the required information as part of knowledgebase for component models and technologies and are responsible for developing the technology parameters that are necessary for the mapping from platform independent models to platform specific models.

In order to adapt TLG to fit in the goal of component specification and code generation, TLG was formally defined as a triple [24]. TLG is used for two purposes in middleware generation, one of which is for the knowledge representation and feature modeling for various aspects of heterogeneous components and the other to support automatic middleware code generation. The configuration knowledge and feature modeling are represented as TLG classes, and the rules for generation are shown as TLG functions. To achieve the automatic middleware generation, the TLG interpreter computes the generation rule functions.

The algorithm for composing heterogeneous components works in the following way: a proxy client is generated for a server component and a proxy server for a client component, the service is then relayed from the original client to proxy server and then to proxy client, which finally reaches the original server object. Meanwhile, a bridge driver is generated to evoke and build a common context between the proxies. This formalism using TLG is yet to be implemented. The UniGGen approach also uses similar kind of algorithm whereas uses templates to achieve the interoperability.

This chapter presented a brief overview of the work related to this project. The next chapter provides an overview of the proposed Glue Generation Framework in UniFrame for the CORBA-Java/RMI Interoperability (UniGGen).

3. UniGGen ARCHITECTURE

In the System generation process of UniFrame, the glue code necessary to bridge the dynamically discovered heterogeneous distributed software components is generated using UniGGen.

3.1 URDS Architecture Overview

An introduction to UniFrame and URDS is provided in section 2.1. This section will provide an overview of the URDS architecture and a description of components comprising it.

The URDS architecture is organized as a federated hierarchy [19]. Every ICB consists of zero or more Headhunters attached to it and the ICBs are in turn linked to one another. The URDS discovery process locates services within an administratively defined logical domain, and domains are determined by the organizations providing the service [19].

The discovery process in URDS is based on multicasting and is designed to handle failures through periodic multicast announcements and information caching. Table 3.1 provides a brief description of each URDS component. Figure 3.1 is an illustration of the URDS architecture and its components.

Internet Component Broker (ICB)	The ICB acts as an all-pervasive component broker in an interconnected environment. It encompasses the communication infrastructure necessary to identify and locate services, enforce domain security and handle mediation between heterogeneous components. The ICB is not a single component, but a collection of services comprising of the Query Manager (QM), the Domain Security Manager (DSM), Adapter Manager (AM), and the Link Manager (LM). These services are reachable at well-known

	addresses. It is envisioned that there will be a fixed number of ICBs deployed at well-known locations hosted by corporations or organizations supporting this initiative.
Domain Security Manager (DSM)	The DSM serves as an authorized third party that handles the secret key generation and distribution and enforces group memberships and access controls to multicast resources through authentication and use of access control lists (ACL). DSM has an associated repository (database) of valid users, passwords, multicast address resources and domains.
Query Manager (QM)	The purpose of the QM is to translate a system integrator's natural language-like query into a structured query language statement and dispatch this query to the 'appropriate' Headhunters, which return the 32 list of service provider components matching these search criteria expressed in the query. 'Appropriate' is determined by the domain of the query. Requests for service components belonging to a specific domain will be dispatched to Headhunters belonging to that domain. The QM, in conjunction with the LM, is also responsible for propagating the queries to other linked ICBs.
Link Manager (LM)	The LM serves to establish links with other ICBs for the purpose of federation and to propagate queries received from the QM to the linked ICBs. The LM is configured by an ICB administrator with the location information of LMs of other ICBs with which links are to be established.
Adapter Manager (AM)	The AM serves as a registry/lookup service for clients seeking adapter components. The adapter components register with the AM and while doing so they indicate their specialization, i.e., which component models they can bridge efficiently. Clients contact the AM to search for adapter components matching their needs.

Headhunter (HH)	The Headhunters perform the following tasks: a) Service Discovery: detect the presence of service providers (Exporters), b) register the functionality of these service providers, and c) return a list of service providers to the ICB that matches the requirements of the component assemblers/system integrators requests forwarded by the QM. The service discovery process performs the search based on multicasting.
Meta-Repository (MR)	The Meta-Repository is a data store that serves a Headhunter to hold the UniFrame specification information of exporters adhering to different models. The repository is implemented as a standard relational database.
S1..Sn	Services implemented in different component models (RMI, CORBA, etc.,) identified by the service type name and the component's informal UniFrame specification which is an XML specification outlining the computational, functional, cooperational and auxiliary attributes of the component and zero or more QoS metrics for the component.
AC1..ACn	Adapter components, which serve as bridges between components implemented in diverse models.
C1..Cn	Component Assemblers, System Integrators, System Developers searching for services matching certain functional and nonfunctional requirements.

Table 3.1 Description of URDS Components (from [19])

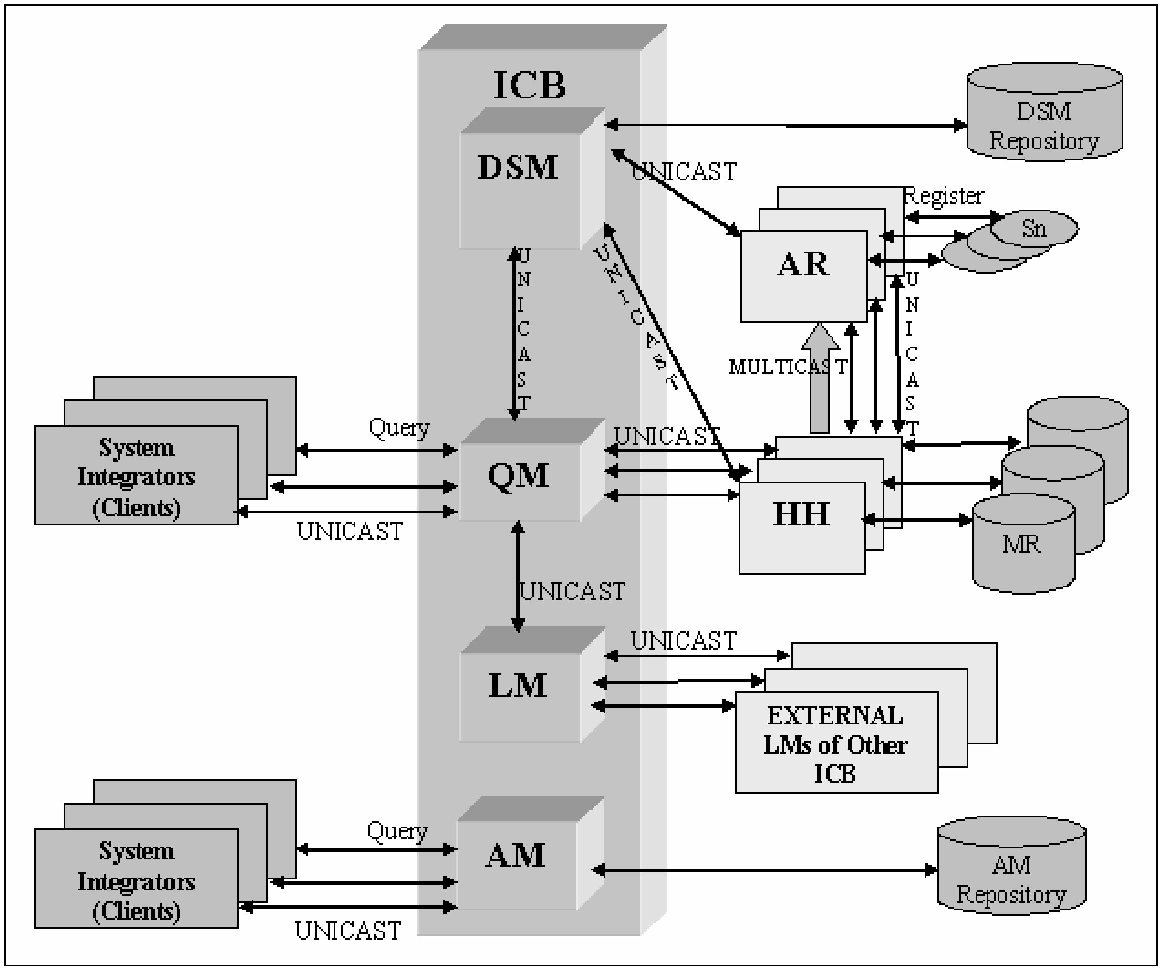


Figure 3.1 URDS architecture (from [19])

The UniGGen architecture automatically generates the glue code required to interoperate between the heterogeneous components that are dynamically discovered by the URDS architecture.

3.2 UniGGen Architecture Details

The UniGGen architecture is designed to provide the infrastructure necessary for semi-automatically generating glue code required to interoperate between heterogeneous (Java RMI and CORBA) components to compose a distributed system, using a template-based approach.

A template is a document or file having a preset format, used as a starting point for a particular application so that the format does not have to be recreated each time it is used. In UniGGen, the template consists of the generic glue code required to interoperate between any Java RMI and CORBA components using RMI-IIOP communication with the parts, such as the interface name of the responder component, in the form of tags ('<' '>'), which are replaced by the UniGGen with appropriate component-specific details during the glue generation process. The code formulated into templates also handles exceptions, for example RemoteException. Figure 3.2 shows a sample code taken from a template.

```
Public class <Glue_CS_RC> extends UnicastRemoteObject implements
<CSInterface_name> {
        private static <CSInterface_name> CS_Ref;
        private static ORB orb;
        private static org.omg.CosNaming.NamingContext root_ctx ;
….
….
        public <Glue_CS_RC>(String s) throws RemoteException {
                super();
        }
….
….
                        <Glue_CS_RC> GCSRC = new <Glue_CS_RC>(Gurl);
                        Naming.rebind(Gurl, GCSRC);
….
….
                        CS_Ref = <CSInterface_name>Helper.narrow(obj);
….
….
….
```

Figure 3.2 Sample template code

In figure 3.2, the part of the code such as CSInterface_name inside “<>” depends on the particular components that are interacting at a point of time. These parts will be replaced by the UniGGen architecture while generating the glue code with the component-specific detail i.e., the name of the interface, which the initiator component requires and the responder component provides.

For software components to be able to interact or interoperate with each other, the components must comply with the rules of their underlying component models. However, two components hosted on different component models have many incompatibility reasons which arise from the differences of the underlying models and the way they present and use software components.

Following are the three basic incompatibilities discussed in [20]:

- ❖ Different Interface Approaches and Implementations: One of the basic elements of an object is its interfaces. Through their interfaces objects expose some details of their functionality. An interface consists of a description of a group of possible operations which a client can ask from an object. A client interacts only with the interfaces of an object, never with the object itself. Interfaces allow objects to appear as black boxes. Different approaches and implementations of objects' interfaces, such as IDL, and Java, make them unknown to clients of other technologies.
- ❖ Different Object References and Storage: When a client wishes to interact with an object, it must first retrieve information, such as the functions or services provided, the parameters that the function accepts and returns about the object's interface. A client's underlying technology must recognize an object's name; it must know where to look and how to retrieve its information, i.e. it must know how the required object's technology stores this information. If a client's technology does not have that ability, then it is impossible for the necessary information of the needed object to be found.
- ❖ Different Protocols: Another basic element in distributed object interactions is the protocols used for the data transmission. The term protocol does not mean only the transport-level protocol, such as TCP/IP, but includes the presentation and session level protocol which the Request Brokers (RBs) support. The transport-level protocol is responsible for the transmission of the data to the end point. The presentation and session level protocols are responsible for the formatting of the

data transmitted from a client to an object, and vice versa, between different RBs.

Table 3.2 presents the basic differences of the two models, Java RMI and CORBA in relation with the above discussed incompatibilities:

Incompatibility points	**CORBA**	**RMI**
Interface Approaches & Implementations	IDL	Java
Object Identification	Identification through Object and Interface Names	Identification through a URL-based Object Name and Interface Name
Object Reference	Reference through an Object Reference (OR)	Reference through a URL-based Object Reference
Object Storage	Storage in Implementation Repository	Storage in rmiregistry
Protocols	GIOP/IIOP/ESIOP	JRMP/IIOP

Table 3.2 CORBA/RMI basic differences in relation with incompatibility points [20]

The differences presented in Table 3.2 are not the only ones between these architectures, but the differences described are the prime causes for incompatibility. The UniGGen proposed in this project tries to solve these differences between the two models.

Here are a few other differences between CORBA and RMI:

- ❖ RMI supports dynamic class downloading whereas CORBA does not.
- ❖ RMI supports dynamic stub downloading whereas CORBA does not.
- ❖ RMI allows passing of objects by value whereas CORBA does not.
- ❖ CORBA supports persistent naming whereas RMI does not.

To interoperate between Java RMI (without IIOP) and CORBA components, a glue code is required for each pair of initiator and responder components which requires a considerable amount of human effort (an initiator is a component that requires services and a responder is a component that provides services in a particular interaction or at the particular instance). The semi-automation performed at runtime using previously formulated generic templates (which contain the required Java RMI-IIOP code) that can be adapted to any number of Java RMI and CORBA components requires less human effort and overhead (in terms of turn around time) on the performance of the system composed.

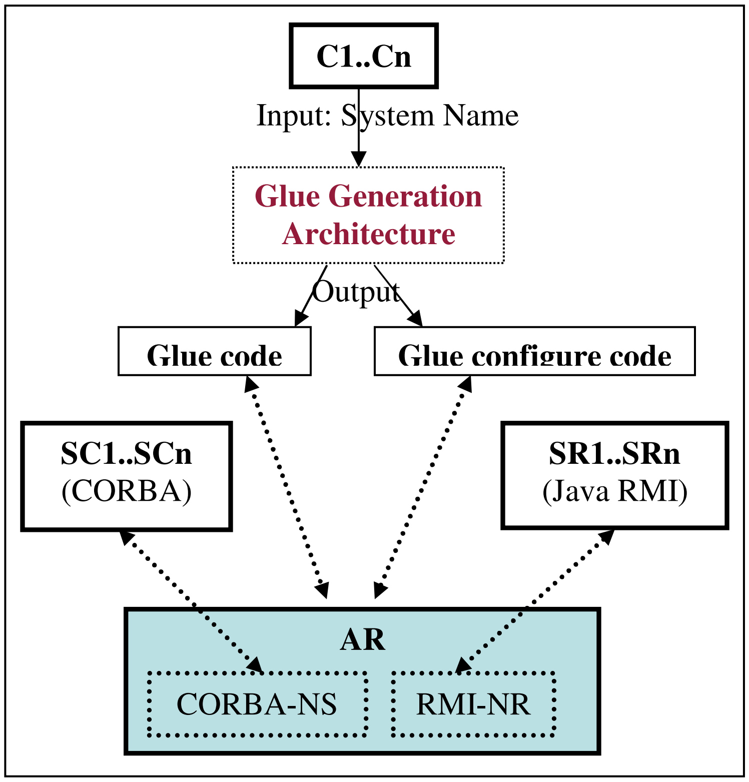


Figure 3.3 UniGGen Architecture

The UniGGen infrastructure (illustrated in Figures 3.3 and 3.4) comprises of the following modules: 1) GlueGeneratorKB (GGKB), 2) GlueGenerator (GG), 3) GlueCodeGen (GCG), 4) GlueConfigureGen (GCfG), 5) GlueCodeGenGUI (GCGGUI), 6) GlueConfigureGenGUI (GCfGGUI), 7) UMMSpecificationParser (UMMSP), and the features of the following URDS components also utilized within the UniGGen design: 8)

Active Registry (AR), 9) S1..Sn. Other URDS components were not utilized within this timeframe, but would be added in the future enhancement. Figure 3.3 also illustrates the users (C1..Cn) of the UniGGen system i.e., the System Integrators.

Here is the brief description of the functionality of each module and how they interact with each other to achieve the functionality of the UniGGen.

- ❖ S1..Sn (SC1..SCn and SR1..SRn together): These are services or components to be composed which are implemented in different component models i.e., CORBA and Java RMI.
- ❖ AR: AR consists of the CORBA-NS (CORBA NamingService), which registers with itself the SC1..SCn components and RMI-R (Java RMI Registry), which registers with itself the SR1..SRn components.
- ❖ C1..Cn: System Integrators, who use the UniGGen architecture to compose a system of components by providing the system name to the Glue Generation Architecture.
- ❖ Glue Generation Architecture: This automates the generations of the required glue code to interoperate between the components, and the glue configure code to configure the initiator component to the glue component and the glue component to the responder component. The generated glue code registers itself with the AR, through which the initiator component can look up the glue component. The generated glue configure code in turn looks up the initiator and the glue component to configure them so as to enable proper communication between initiator and responder components through glue. Figure 3.4 elaborates the Glue Generation Architecture.
- ❖ GlueGeneratorKB: This module consists of the information required by the glue generation process such as the UMM specification URL's of the components to be composed and their interactions. The information contained in the knowledgebase is separated from the actual process of glue generation because when new systems need to be added in the future, the only thing that needs to be updated and maintained is the knowledgebase.
- ❖ GlueGenerator: This module gets the UMMSpecTable and the

ComponentInteractionTable from GlueGeneratorKB for the System name obtained from the C1..Cn (users). For each component interaction pair, it passes the UMM specification URL's of both initiator and responder components to UMMSpecificationParser and gets back the component details. GlueGenerator then retrieves the technologies of both the components from these details and if the technologies are different i.e., Java RMI and CORBA, then it passes the component details to GlueCodeGen and GlueConfigureGen.

- GlueCodeGen: This module retrieves the details, such as the interface name of the responder component, and generates the glue code using the glue code template.
- GlueCodeGenGUI: This module compiles and deploys the generated glue code.
- GlueConfigureGen: This module retrieves the details, such as the interface name of the responder component, and generates the glue configure code using the glue configure template.
- GlueConfigureGenGUI: This module compiles and deploys the generated glue configure code.
- UMMSpecificationParser: The UMMSpecificationParser parses the contents of the XML document containing UMM specifications to retrieve the required details and returns them to the GlueGenerator.

Figure 3.4 shows the flow of activities of Glue Generation Architecture.

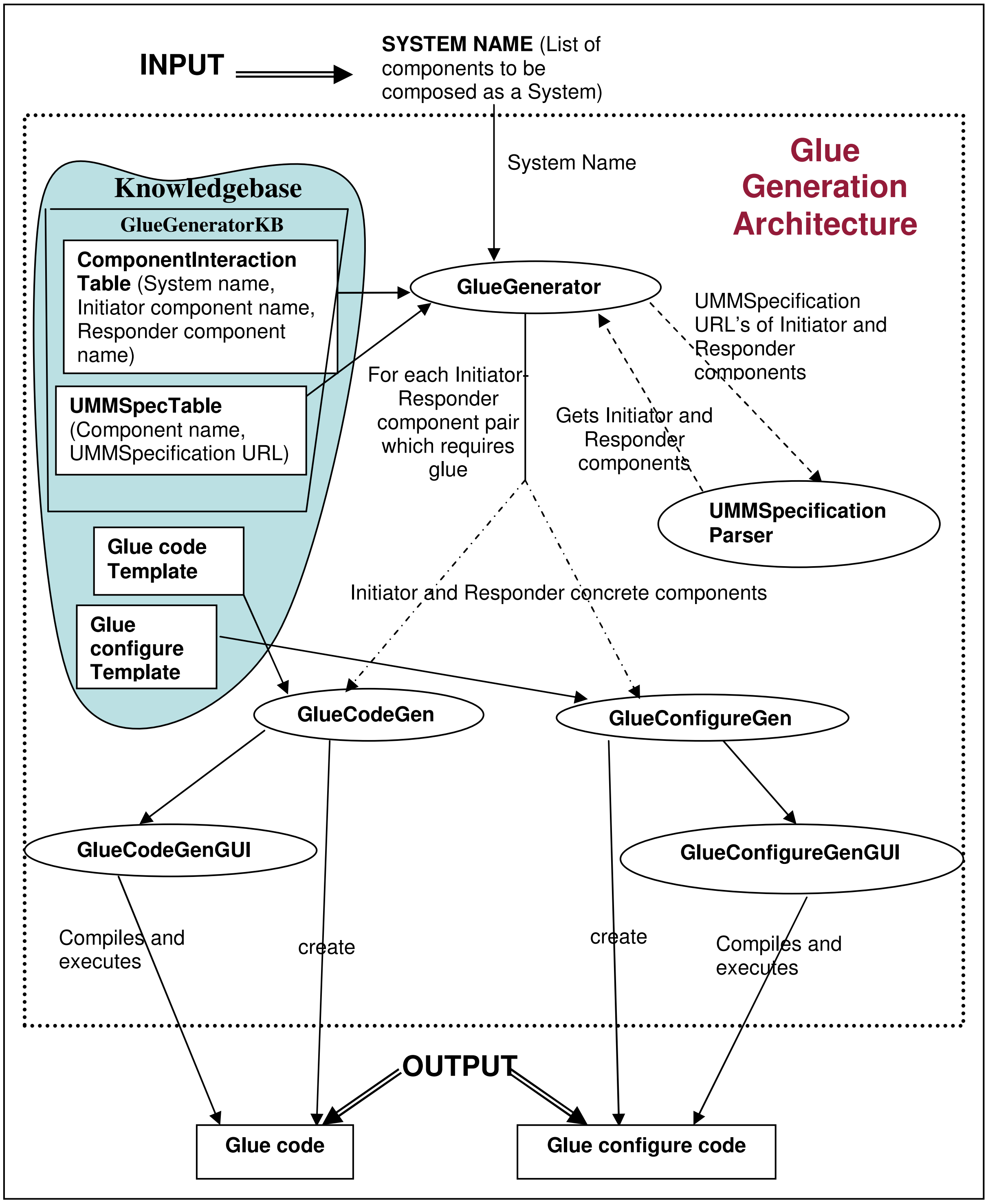

INPUT
SYSTEM NAME (List of components to be composed as a System)
System Name
Glue Generation Architecture
Knowledgebase
GlueGeneratorKB
ComponentInteraction Table (System name, Initiator component name, Responder component name)
UMMSpecTable (Component name, UMMSpecification URL)
Glue code Template
Glue configure Template
GlueGenerator
UMMSpecification URL's of Initiator and Responder components
For each Initiator-Responder component pair which requires glue
Gets Initiator and Responder components
UMMSpecification Parser
Initiator and Responder concrete components
GlueCodeGen
GlueConfigureGen
GlueCodeGenGUI
GlueConfigureGenGUI
Compiles and executes
create
create
Compiles and executes
OUTPUT
Glue code
Glue configure code

Figure 3.4 Elaboration of the Glue Generation Architecture shown in Fig 3.3

The detailed algorithms for each module are discussed in Section 3.3. Here is a brief description of the flow of activities as shown in Figure 3.4:

The GlueGenerator accepts the name of the System to be composed from the Client and then retrieves the URL of the XML documents consisting of the UMM specifications of the components along with the interactions between components forming the System from the GlueGeneratorKB. Then, the GlueGenerator passes the UMM specification URL's of initiator and responder components of each interaction to UMMSpecificationParser. UMMSpecificationParser parses these XML documents, creates componentEntity's by setting the attributes and returns them to the GlueGenerator. The GlueGenerator then retrieves the technology (underlying object model) in which the initiator and responder components of each interaction are implemented, checks if they are different. If the technologies are different, the GlueGenerator passes the initiator and responder components to the GlueCodeGen and GlueConfigureGen. These are two independent activities.

The GlueCodeGen reads from the "Glue code Template" (present in the knowledgebase), inserts the component-specific details, such as interface name, services offered in the placeholders provided in the template and writes it to a Java file (containing the required glue code). Similarly, the GlueConfigureGen reads from the "Glue configure Template", inserts the component-specific details in the placeholders provided in the template and writes it to a Java file (containing the required glue configure code). The GlueCodeGenGUI and GlueConfigureGenGUI deploy the generated "Glue code" and "Glue configure code".

3.3 Modules of UniGGen

This section describes the functionality of each module in the UniGGen. Table 3.3 shows the data structures used in the algorithms by these modules.

Hash table UMMSpecTable	Mapping between component name and URL of the UMM specification xml document.
Hash table ComponentInteractionTable	Mapping between initiator component name and responder component name of a single system.
Hash table SystemTable	Mapping between System name and above mentioned ComponentInteractionTable.
Entity componentEntity	An entity, that can be stored in a persistent repository and holds all the attributes corresponding to a UniFrame component specification, such as interface name, technology, method signature in the form of methodEntity.
Entity methodEntity	An entity, that can be stored in a persistent repository and holds all the attributes corresponding to a UniFrame method specification such as method name, method return type, parameters signatures i.e., order of parameters.
Entity parameterEntity	An entity, that can be stored in a persistent repository and holds all the attributes corresponding to a UniFrame parameter specification such as parameter name, parameter return type.

Table 3.3 Data Structures used in the algorithms by the UniGGen modules

3.3.1 GlueGeneratorKB (GGKB)

The GlueGeneratorKB consists of the KnowledgeBase (KB) required for the UniGGen architecture. It provides the information of family of the systems that the UniGGen requires for the glue generation process in the form of UMMSpecTable and ComponentInteractionTable. The UMMSpecTable consists of a list of XML documents containing the UMM specifications of the components forming the family of systems. The ComponentInteractionTable consists of the interactions between the components that are part of the family of systems the UniGGen can compose. This KB is created beforehand or can be augmented when a new system becomes part of the family of systems already known. The data structures used to store this information are hash tables, which can be easily created and modified.

3.3.2 GlueGenerator (GG)

The GlueGenerator is responsible for checking the incompatibility, i.e., the difference in the technology or object models between initiator-responder component pairs, and generating the required glue code. Here is the step-by-step process for checking the incompatibilities and generating the glue code:

1. The GlueGenerator obtains the name of the System in the form of a string to be composed from the User or System Integrator.
2. The GlueGenerator obtains the UMMSpecTable, and ComponentInteractionTable for the System to be composed from the GlueGeneratorKB.
3. For each initiator-responder pair existing in ComponentInteractionTable for the System to be composed, the UMM specification URL of the XML document for both components are passed to the UMMSpecificationPasrser which parses the XML document and returns an instance of componentEntity with details such as component name, provided interface name and a list of services or operations the component provides and their signature, required interface name and a list of services or operations that the component needs or requires and their signatures,

implementation technology, and with which name the component registers itself with the Active Registry (i.e., the name with which CORBA components register themselves in CORBA NamingService and Java RMI components in RMI Registry).

4. Next, GlueGenerator checks if the implementation technology retrieved from the componentEntity's of both initiator and responder components are different i.e., Java RMI and CORBA. If the technologies are different, then the instances of the componentEntity's of both initiator and responder are passed to the instances of GlueCodeGen and GlueConfigureGen otherwise, it does nothing thereby, indicating that glue code is not required.

The Section 3.3.2.1 provides this process in the form of an algorithm.

3.3.2.1 Algorithm for checking the incompatibility between initiator and responder components to generate the required glue code

This algorithm provides the process for checking if there exists a difference in technology between initiator and responder components. For each initiator-responder pair, the GlueGenerator passes the initiator and responder's XML-based UniFrame specification to UMMSpecificationParser and obtains Entity objects, initiatorcomponentEntity and respondercomponentEntity. The GlueGenerator then uses these componentEntity's to determine the technology in which each component of an interaction is implemented and then checks to see if the technologies match. If the technologies do not match (it means one of the initiator and responder components is in Java RMI and the other is in CORBA), then the GlueGenerator creates instances of GlueCodeGen and GlueConfigureGen, passing to them the initiator and responder componentEntity's to generate the glue code required to interoperate between that initiator-responder component pair.

```
GG_CHECK_INCOMPATIBILITY
    //Obtain the System name
    systemName = GET name of the System
    //Obtain a list of the locations (URL) of the UniFrame Specification
    //for the components of the systemName from GGKB
    UMMSpecTable = GET from UMMSpecTable of GGKB for systemName
    in SystemTable
    //Obtain a list of the initiator-responder pairs of the components of //the
    systemName from GGKB
    ComponentInteractionTable = GET from ComponentInteractionTable of
    GGKB for systemName in SystemTable
    FOR EACH componentInteraction of ComponentInteractionTable
        CREATE a initiatorcomponentEntity
        CREATE a respondercomponentEntity
        //Parse the UniFrame Specification and construct a
    componentEntity
        initiatorcomponentEntity = GG_PARSE_UNIFRAME_SPEC
(Section 3.3.7.1) with UniFrame Specification URL of initiator
        initiatorcomponentTechnology = GET componentTechnology
    from initiatorcomponentEntity
        //GET the technologies of initiator and responder
        //from their respective componentEntity's
        respondercomponentEntity = GG_PARSE_UNIFRAME_SPEC
    (Section 3.3.7.1) with UniFrame Specification URL of responder
        respondercomponentTechnology = GET componentTechnology
    from respondercomponentEntity
        IF initiatorcomponentTechnology NOT EQUALS
    respondercomponentTechnology
            //Call GCG and GCfG to generate glue code and glue
            //configure code with initiator and responder
            //componentEntity's
```

```
                CALL GCG_GENERATE_GLUE_CODE (Section 3.3.3.1)
                CALL GCfG_GENERATE_GLUE_CONFIGURE_CODE (Section 3.3.4.1)
            ENDIF
        ENDFOR
END_GG_CHECK_INCOMPATIBILITY
```

3.3.3 GlueCodeGen (GCG)

The GlueCodeGen retrieves the details of the initiator and responder components, for example the required interface of the initiator, the provided interface of the responder and the functional details, such as the function name, the signature, parameters from the componentEntity's and then reads the text from the glue code template and writes the glue code into a Java file after replacing the tags with the appropriate details. The GlueCodeGen assumes that the Syntactic contract of the services (functions) required by the initiator and services (functions) provided by the responder are same for an interaction. The glue code (.java) file created consists of the Java RMI-IIOP code required to communicate between the particular initiator and responder components. The glue code registers itself in the AR for the initiator component to be able to communicate with it. Next, GlueCodeGen creates an instance of GlueCodeGenGUI.

3.3.3.1 Algorithm for generating glue code

This algorithm outlines the process of generating the glue code required to interoperate between the initiator and responder component. It gets the interface and functional details such as the Syntactic Contract from the initiator and responder componentEntity's, reads the template text file and writes to a Java file with the tags, such as "InterfaceName", and "MethodName", replaced with attributes in componentEntity's that were populated using UMMSpecificationParser. It then calls GlueCodeGenGUI to deploy the generated glue code.

```
GCG_GENERATE_GLUE_CODE
        IN: initiatorcomponentEntity and respondercomponentEntity
        CREATE Java file
        READ glue code template file
        IF text read from template file is not a tag
                WRITE template code into Java file as it is
        ELSE IF tag in template EQUALS attribute of initiatorcomponentEntity
                GET attribute value from componentEntity
                SET tag to attribute value and WRITE tag value to generated Java file
        ENDIF
        CALL GCGGUI_DEPLOY_GLUE (Section 3.3.5.1)
END_GCG_GENERATE_GLUE_CODE
```

3.3.4 GlueConfigureGen (GCfG)

GlueConfigureGen also retrieves the details of the initiator and responder components from the Component instances, reads the text from the glue configure code template and writes the code into a Java file after replacing the tags with the appropriate details. The glue configure code (.java) file created consists of the Java RMI-IIOP code required to configure the glue to the responder component and also to configure the initiator component to the glue component i.e., configure so as the initiator will be able to communicate with responder component through glue component by obtaining the reference of the glue component. Next, GlueConfigureGen creates an instance of GlueConfigureGenGUI to deploy the generated glue configure code.

3.3.4.1 Algorithm for generating glue configure code

This algorithm outlines the process of generating the glue configure code required to configure the glue code for it to be able to communicate with responder component. It gets the interface name and the name with which the responder component registers itself in AR, reads the template text file and writes to a Java file with the tags, such as “InterfaceName”, “ResponderLocation” replaced with attributes in componentEntity’s that were populated using UMMSpecificationParser. It then calls GlueConfigureGenGUI to deploy the generated glue configure code.

```
GCfG_GENERATE_GLUE_CONFIGURE_CODE
    IN: initiatorcomponentEntity and respondercomponentEntity
    CREATE Java file
    READ glue configure code template file
    IF text read from template file is not a tag
        WRITE template code into Java file as it is
    ELSE IF tag in template EQUALS attribute of initiatorcomponentEntity
        GET attribute value from componentEntity
        SET tag to attribute value and WRITE tag value to generated Java file
    ENDIF
    CALL GCfGGUI_DEPLOY_GLUE (Section 3.3.6.1)
END_ GCfG_GENERATE_GLUE_CODE
```

3.3.5 GlueCodeGenGUI (GCGGUI)

The GlueCodeGenGUI compiles and executes/deploys the generated glue component (Java file).

3.3.5.1 Algorithm for deploying the generated glue code

This algorithm outlines the process of deploying the glue code.

```
GCGGUI_DEPLOY_GLUE
    IN: Generated Glue code file name
    COMPILE Glue Code
    EXECUTE Glue Code
```

```
END_ GCGGUI_DEPLOY_GLUE
```

3.3.6 GlueConfigureGenGUI (GCfGGUI)

The GlueConfigureGenGUI compiles and executes/deploys the generated glue configure component (Java file).

3.3.6.1 Algorithm for deploying the generated glue configure code

This algorithm outlines the process of deploying the glue configure code.

```
GCfGGUI_DEPLOY_GLUE
    IN: Generated Glue configure code file name
    COMPILE Glue Configure Code
    EXECUTE Glue Configure Code
END_ GCfGGUI_DEPLOY_GLUE
```

3.3.7 UMMSpecificationParser (UMMSP)

The UMMSpecificationParser parses the contents of XML document containing UMMSpecifications to retrieve the required details such as the provided and the required interface names, signatures of functions to be called from the provided interface of responder and the required interface of initiator and creates an instance of componentEntity of both initiator and responder.

3.3.7.1 Algorithm for Parsing the UniFrame Specification

This algorithm uses recursion to parse through the nodes of the XML tree (present in the UniFrame Specification URL), extract the node values and store these values in the componentEntity. The algorithm starts parsing at the root node element. It extracts the node name and checks if the node name matches any attribute in componentEntity and populates it with the value in this node. It then finds all the children of that node and repeats the process through recursion. In this process, it populates the numberofMethods, the methodEntity consisting of the method signature, numberofParameters and parameters' signatures in each methodEntity.

```
GG_PARSE_UNIFRAME_SPEC
    IN: UniFrame Specification URL
    OUT: componentEntity
    nodeName = GET NODE NAME from nodeElement
    FOR each attribute in componentEntity
        IF nodeName EQUALS attribute
            nodeValue = GET NODE VALUE from nodeElement
            SET attribute value in componentEntity to nodeValue
        ENDIF
    ENDFOR
    //Get the list of children for this Node.
    NODELIST childrenList = SET CHILDNODES for nodeElement
    IF childrenList NOT NULL
        //For every child node in the list
        FOR I = 0 to LENGTH of childrenList
            childNode = childrenList[i]
            //Recurse through the function READ_NODE passing it the
            //childNode as reference
            CALL GG_PARSE_UNIFRAME_SPEC with childNode, which
            will SET attribute values in methodEntity that in turn consist of
            attribute values for parameterEntity
        ENDFOR
    ENDIF
END_ GG_PARSE_UNIFRAME_SPEC
```

3.3.8 Active Registry (AR)

The Active Registry in the UniGGen is the Java RMI Naming Registry (RMI-R) and CORBA NamingService (CORBA-NS) itself. RMI-R and CORBA-NS each consists of a list of all the components registered in them and the ability to look up and allow other components to connect to and communicate with the services offered by the components registered within them. The use of the registries is efficient as it is relatively easy and quick to register a component for external Internet access and greatly simplifies the process of Active Registration.

3.3.9 S1..Sn

Services implemented in different component models (RMI, CORBA) identified by the service type name and the component's UniFrame specification, which is an XML specification outlining the functional attributes of the component and the nonfunctional attributes, such as cooperational attributes, are assumed to be already known. This information present in the KnowledgeBase of UniGGen is to be actually passed to it from the System Generator or Integrator (C1..Cn) at the time of composing the system to check if any incompatibilities are present between the components and try to integrate them if possible.

The UniGGen architecture takes care of two way communication between Java RMI and CORBA components, i.e., a function call or request-reply communication between a Java RMI component on a CORBA component and vice-versa.

3.4 Architecture for the placement of the generated glue components

Now that glue components required for each initiator-responder component pairs have been created, deployed and configured so that all components can communicate with each other, the System composed of the individual components is ready.

The performance of the composed System measured as the response time (i.e., the time taken for the initiator to request a service and receive reply from responder through a glue component) depends upon where the glue components are deployed. A few choices for the placement of the generated glue code are:

- Centralized - All the glue components are placed and deployed in a centralized manner, i.e., on a single machine where the System Integrator or user is executing the UniGGen. This architecture will have a significant amount of overhead on the performance of the composed System as every time the initiator component needs to communicate with the responder component, the call has to pass through the glue component deployed on a central machine, which may become a bottleneck.
- Distributed - The glue components are placed and deployed in a distributed manner i.e., glue components are deployed on different machines. A few choices for distributed placement of glue code are:
    - Each glue component required for the initiator component is deployed on the same machine as that of the initiator component: This architecture will have less overhead on the performance of the System compared to the centralized placement of the glue components as the call from initiator component to responder component will pass through the glue component on the same machine as that of initiator component and not have to go through any extra machines.
    - Each glue component required for the initiator component is deployed on same machine as respective responder component: This architecture will have similar overhead on the performance of the System as the above architecture for non-centralized placement of the glue components as the call from initiator component to responder component will pass through

the glue component on the same machine as the responder component, which is similar to placement on the initiator machine since the call does not have to pass through any extra machines.

- Each glue component is deployed randomly on different machines: This architecture will have more overhead on the performance of the System compared to the above distributed architectures as the call from initiator component to responder component may have to pass through the glue component on a machine that is neither the machine where initiator component nor responder component are deployed.

A few issues related to environment and access permissions that can arise are:

- ❖ Difference in operating system: The operating system installed on the machine where the glue code is to be placed and deployed may not have the proper architectural support.
- ❖ The software required to enable proper execution of the generated glue code may not be present.
- ❖ The machines should be able to support file transfer.
- ❖ The user may require special access permissions such as “execute” permission from the Administrator.

Of the above four choices proposed for the placement of the generated glue components when the issues related to environment and access permissions are not considered, the distributed architecture that deploys the glue components on the same machines as that of the initiator or responder components will have the least overhead on the performance of the System composed. Experiments were carried out to validate the proposed UniGGen architecture and the choices for placement of the generated glue code and the results obtained are discussed in chapter 5.

4. IMPLEMENTATION OF THE UniGGen ARCHITECTURE

The previous chapter described the conceptual perspective of the UniGGen architecture and provided choices for placement of the generated glue code. This Chapter describes the prototype implementation of UniGGen using Java and Java based technologies.

4.1 Technology

This section describes various architectural artifacts and technologies that have been leveraged in the implementation the UniGGen architecture.

4.1.1 Java^{TM} 2 Platform Enterprise Edition (J2EE^{TM})

The prototype implementation is based on the architectural model laid out by [21] in J2EE. J2EE defines a standard that applies to all aspects of architecting, developing, and deploying multi-tier, server-based applications. Figure 4.1 [from 22] shows the components of the J2EE Model.

The J2EE platform specifies technologies to support multi-tier enterprise applications. These technologies fall into three categories [21]: Component, Service, and Communication. The following sub-sections outline the technologies (see Figure 4.1) in each of these categories that have been used in the implementation. (The Java based technologies described in these sections are proprietary technologies of SUN Microsystems [21]).

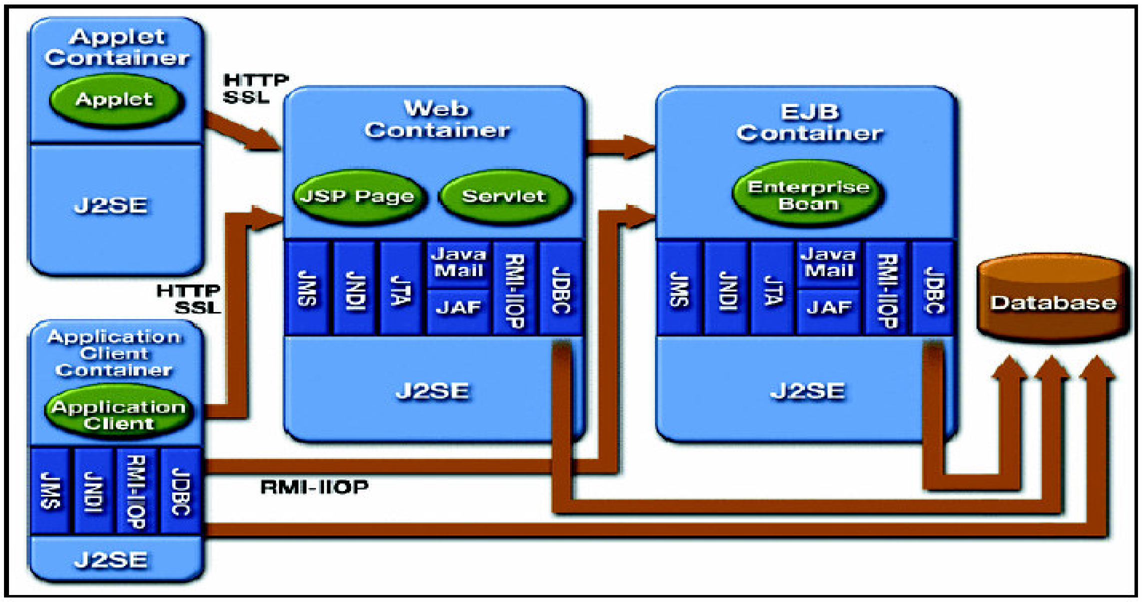


Figure 4.1 Components and Containers of J2EE Model.

(Figure 4.1 is courtesy of SUN Microsystems, [22]).

4.1.1.1 Component Technologies

The J2EE Component technologies have been used in the prototype to create the front-end client components and back-end service components. The prototype supports Application Clients which are structured as Application Client Components (described in section 4.1.1.1.1).

The different types of J2EE components used in the prototype are described below.

4.1.1.1.1 Application Client Components

Application clients are client components that execute in their own Java virtual machine. Application clients are implemented using Project Swing, the part of the Java^{TM} Foundation Classes (JFC) software that implements a set of GUI components with a pluggable look and feel. Project Swing is implemented entirely in the Java programming language, and is based on the JDK^{TM} 1.1 Lightweight UI Framework.

4.1.1.2 Service Technologies

The J2EE platform service technologies allow applications to access a variety of services. The prominent service technologies supported are JDBCTM API 2.0, which provides access to databases, Java Transaction API (JTA) 1.0 for transaction processing, Java Naming and Directory Interface (JNDI) 1.2, which provides access to naming and directory services, and J2EE Connector Architecture 1.0, which supports access to enterprise information systems.

The service technologies used in the prototype are described below:

❖ Java API for XML Processing 1.1: XML is a language for representing text based data so the data can be read and handled by any program or tool. Programs and tools can generate XML documents that other programs and tools can read and handle. Java API for XML Processing (JAXP) supports processing of XML documents using DOM, SAX, and XSLT parsers. JAXP enables applications to parse and transform XML documents independent of a particular XML processing implementation.

4.1.1.3 Communication Technologies

Communication technologies provide mechanisms for communication between clients and servers and between collaborating objects hosted by different servers. Some of the communications technologies supported by the J2EE Platform include – Transport Control Protocol over Internet Protocol (TCP/IP), Hypertext Transfer Protocol HTTP 1.0, Secure Socket Layer SSL 3.0, Java Remote Method Protocol (JRMP), Java IDL, Remote Method Invocation over Internet Inter ORB Protocol (RMI-IIOP), Java Message Service 1.0 (JMS), JavaMail and Java Activation Framework.

The inter-component communication on the server side is achieved through Java and the HTTP 1.0.

4.2 Prototype Implementation

This section describes the implementation of the prototype for the UniGGen architecture described in Chapter 3.

The prototype includes the application clients (GlueGeneratorGUI, GlueCompGUI, GlueCompInteractionsGUI, SysArchGUI, GlueQoSGUI) as Java Swing classes along with the component services (GlueGenerator, GlueCodeGen, GlueConfigureGen, GlueCodeGenGUI, GlueConfigureGenGUI, UMMSpecificationParser which form the core of the UniGGen architecture) as Java based services.

4.2.1 Environment

In the prototype, the algorithms outlined for the various components are implemented using the JavaTM 2 Platform, Standard Edition (J2SE) [23] version 1.4 software environment.

The glue code generated executes on an orb2 platform, a set of CORBA-based software tools designed for the creation and deployment of distributed applications, provided by 2AB ORB (Object Request Broker) architecture [4]. The orb2 provides the ability to communicate, share information and create independent layers of logic untouched by changes to operational systems - these benefits and more can and will be available when a properly executed ORB strategy has been put into place. orb2 enables the user to design, implement, operate, manage and maintain distributed applications that run on any combination of Windows or UNIX, proprietary midrange systems or mainframes.

4.2.2 Programming Model

In the implementation of the prototype, the UniGGen functionality is partitioned into modules, and these modules are decomposed into specific objects to represent the behavior and data of the application.

4.2.2.1 Service Components

A service component here refers to a software unit that provides a service. The service provided could be a computational effort or an access to underlying resources. A service component consists of one or more artifacts (software, hardware, libraries) that are integrated together to provide the service.

The service components implemented in the prototype are the GlueGenerator, GlueCodeGen, GlueConfigureGen, GlueCodeGenGUI, GlueConfigureGenGUI, and UMMSpecificationParser. These service components utilize the Component, Method, and Parameter classes to achieve their functionalities and store and retrieve information.

The data structures and the service components of the prototype are illustrated using class diagrams in Appendices A and B.

4.2.2.2 User Interface

A user-friendly interface is provided for the System Integrator or developer to be able to use the prototype easily. The following are the application client components, which present an interface to the user: GlueGeneratorGUI (see Figure 4.3), GlueCompGUI (see Figure 4.4), GlueCompInteractionsGUI (see Figure 4.5), SysArchGUI (see Figures 4.6, 4.7, & 4.8), GlueQoSGUI (see Figures 4.9 & 4.10). Figure 4.2 illustrates the flow of activities between these components.

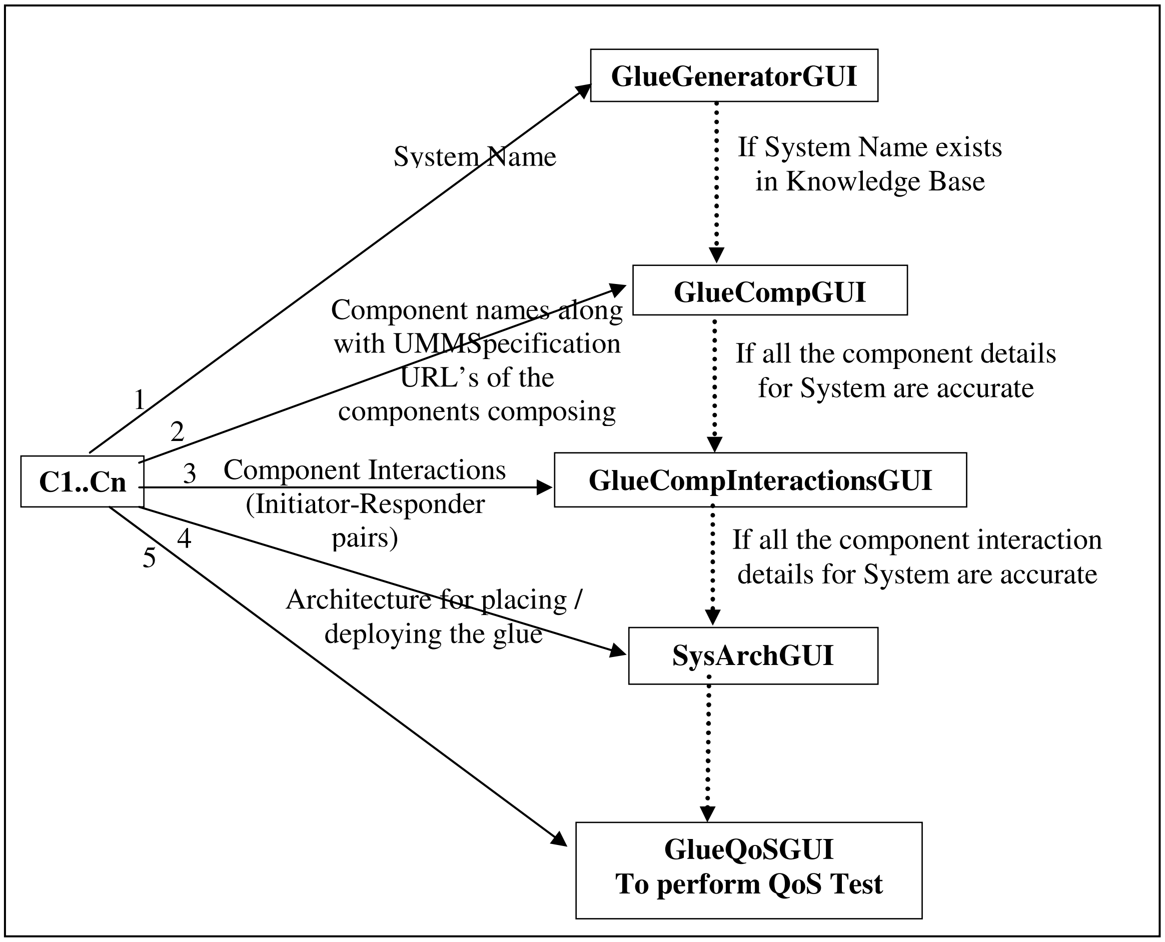
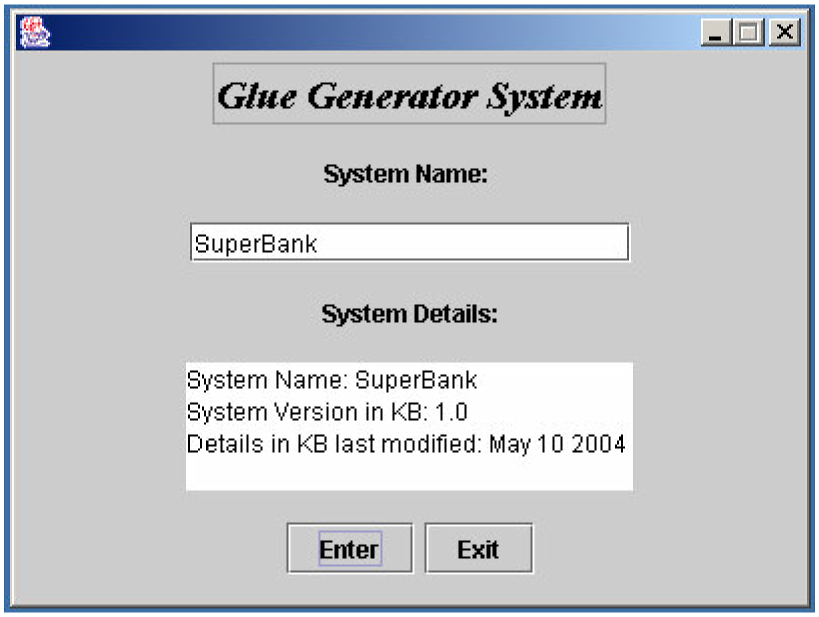

GlueGeneratorGUI
System Name
If System Name exists in Knowledge Base
GlueCompGUI
Component names along with UMMSpecification URL's of the components composing
If all the component details for System are accurate
1
2
C1..Cn
3
Component Interactions (Initiator-Responder pairs)
GlueCompInteractionsGUI
4
5
If all the component interaction details for System are accurate
Architecture for placing / deploying the glue
SysArchGUI
GlueQoSGUI To perform QoS Test

Figure 4.2 UniGGen User Interface Implementation

The functionality associated with these interface components is explained below.

GlueGeneratorGUI: The GlueGeneratorGUI is the user-friendly front-end for the UniGGen prototype. It accepts the name of the System to be composed by the UniGGen prototype and when "ENTER" button is clicked, checks whether this System is present in the Knowledgebase. If this System is known to the Knowledgebase, then it shows the details of the system in the text area provided such as the version and the last modified date in Knowledgebase and creates an instance of GlueCompGUI, otherwise it doesn't proceed further and lets the user know that the System name is invalid. Figure 4.3 illustrates the view of GlueGeneratorGUI with the system name entered as "SuperBank". It also shows the details of the "SuperBank" system in the text area.



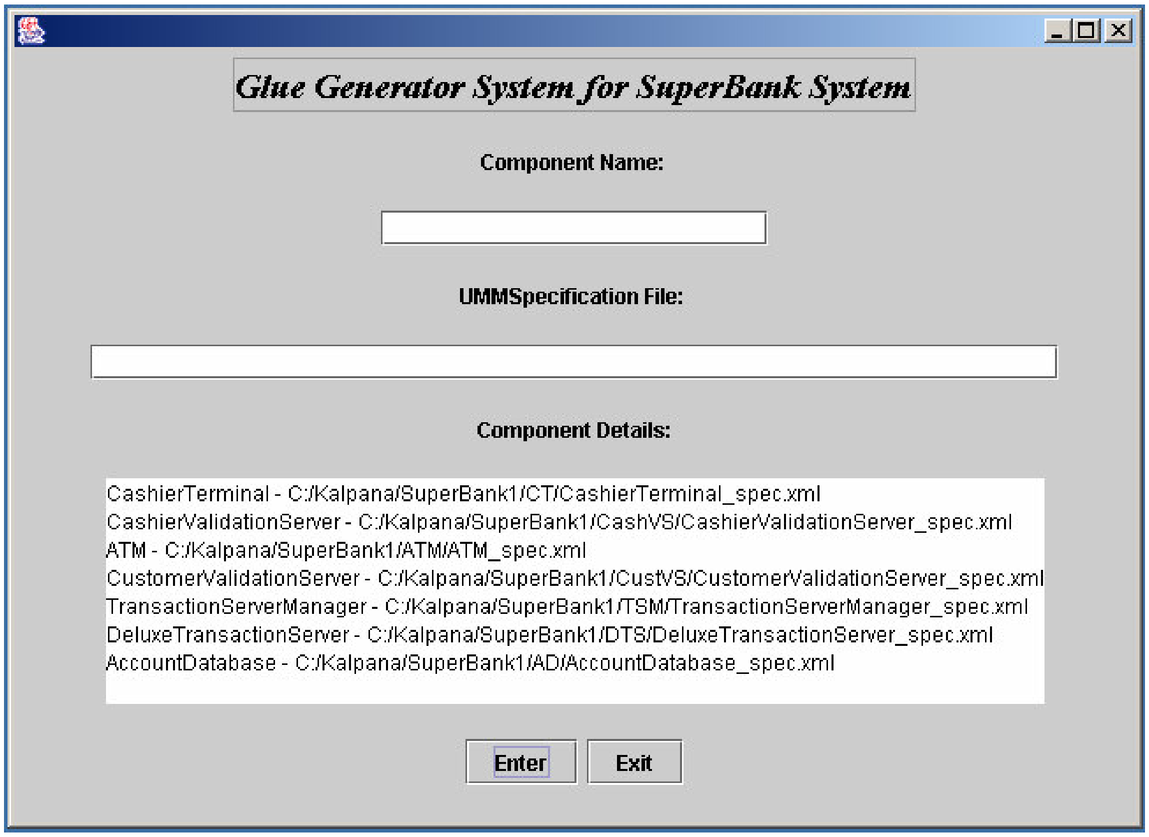
SuperBank

System Details:

System Name: SuperBank
System Version in KB: 1.0
Details in KB last modified: May 10 2004

Figure 4.3 GlueGeneratorGUI view

GlueCompGUI: The GlueCompGUI is a user-friendly interface, which accepts a list of components along with the URL's of the XML documents consisting of UMMSpecifications of the System name accepted by GlueGeneratorGUI. When all the component details are entered, it checks whether the details match those present in the Knowledgebase. If the details match, it shows the accurate details in the text area available in the form of "component name – URL of XML document containing the UMM specifications". Then it creates an instance of GlueCompInteractionsGUI; otherwise it doesn't proceed further and lets the user know that the details are invalid. Figure 4.4 illustrates the view of GlueCompGUI. It shows the details of the 7 components composing the "SuperBank" system.

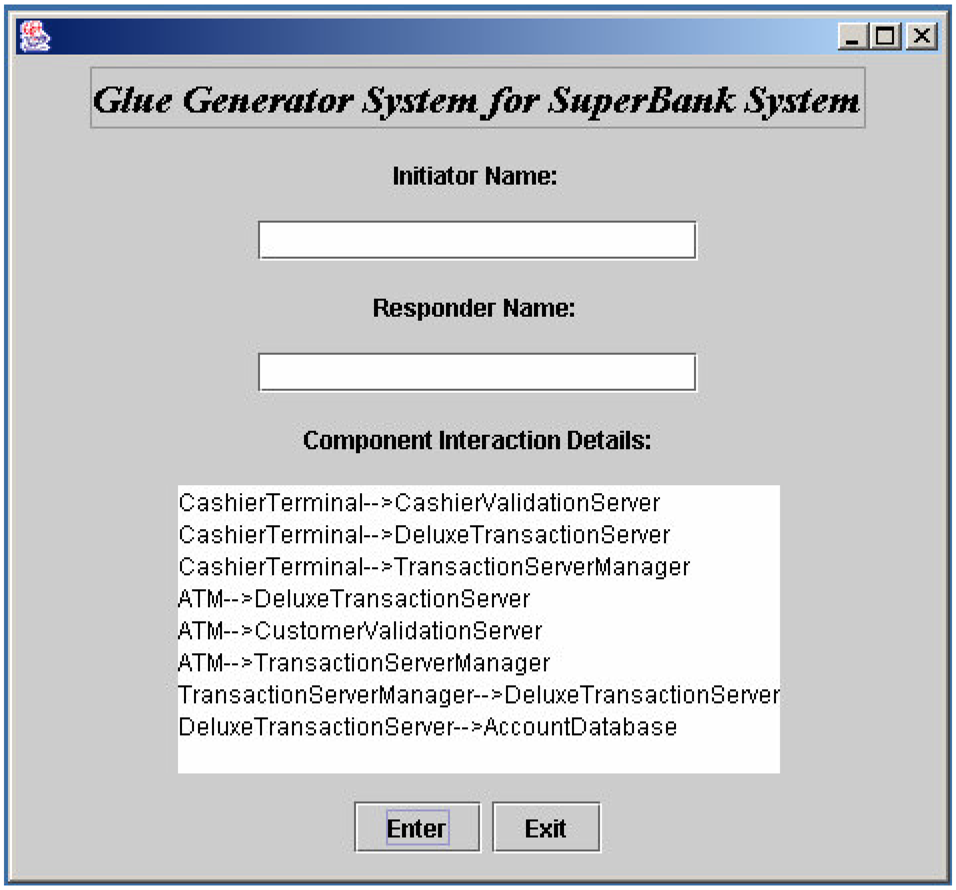


```
CashierTerminal - C:/Kalpana/SuperBank1/CT/CashierTerminal_spec.xml
CashierValidationServer - C:/Kalpana/SuperBank1/CashVS/CashierValidationServer_spec.xml
ATM - C:/Kalpana/SuperBank1/ATM/ATM_spec.xml
CustomerValidationServer - C:/Kalpana/SuperBank1/CustVS/CustomerValidationServer_spec.xml
TransactionServerManager - C:/Kalpana/SuperBank1/TSM/TransactionServerManager_spec.xml
DeluxeTransactionServer - C:/Kalpana/SuperBank1/DTS/DeluxeTransactionServer_spec.xml
AccountDatabase - C:/Kalpana/SuperBank1/AD/AccountDatabase_spec.xml
```

Figure 4.4 GlueCompGUI view

GlueCompInteractionsGUI: The GlueCompInteractionsGUI does a task similar to that of GlueCompGUI in that it accepts a list of component interactions (i.e., initiator-responder component pairs) for the System accepted by GlueGeneratorGUI and checks whether the interaction details match those obtained from the KB. If the details match, it shows the accurate details in the text area available in the form of "initiator component name --> responder component name", and then an instance of GlueQoSGUI is created. Otherwise it does not proceed further and lets the user know that the details are invalid. Figure 4.5 illustrates the view of GlueCompInteractionsGUI. It shows the details of the eight interactions between components composing the "SuperBank" system. For example, "CashierTerminal-->TransactionServerManager" means that the "CashierTerminal" component uses the services of "TransactionServerManager" and "TransactionServerManager-->DeluxeTransactionServer" means that the "TransactionServerManager" component uses the services of

"DeluxeTransactionServer". Although, these two interactions imply that there exists an indirect interaction "CashierTerminal-->DeluxeTransactionServer", this is explicitly shown since the "CashierTerminal" component directly uses "DeluxeTransactionServer" for few services.


Glue Generator System for SuperBank System
Initiator Name:
Responder Name:
Component Interaction Details:
Enter
Exit

```
CashierTerminal-->CashierValidationServer
CashierTerminal-->DeluxeTransactionServer
CashierTerminal-->TransactionServerManager
ATM-->DeluxeTransactionServer
ATM-->CustomerValidationServer
ATM-->TransactionServerManager
TransactionServerManager-->DeluxeTransactionServer
DeluxeTransactionServer-->AccountDatabase
```

Figure 4.5 GlueCompInteractionsGUI view

SysArchGUI: The SysArchGUI is a user-friendly interface through which it provides the System Integrator/user options to choose for placement of Glue, i.e., the categories provided in 3.4. After choosing, the System Integrator can see the System Architecture details along with the interactions which are incompatible or that require glue by clicking on "ShowSystemArchitecture" button and then the user can click on the "SystemGeneration" button which creates an instance of GlueGenerator to start the Glue generation. Figures 4.6, 4.7, & 4.8 illustrate the views of SysArchGUI. Figure 4.6 shows the choices provided for placement of glue for the “SuperBank” system.

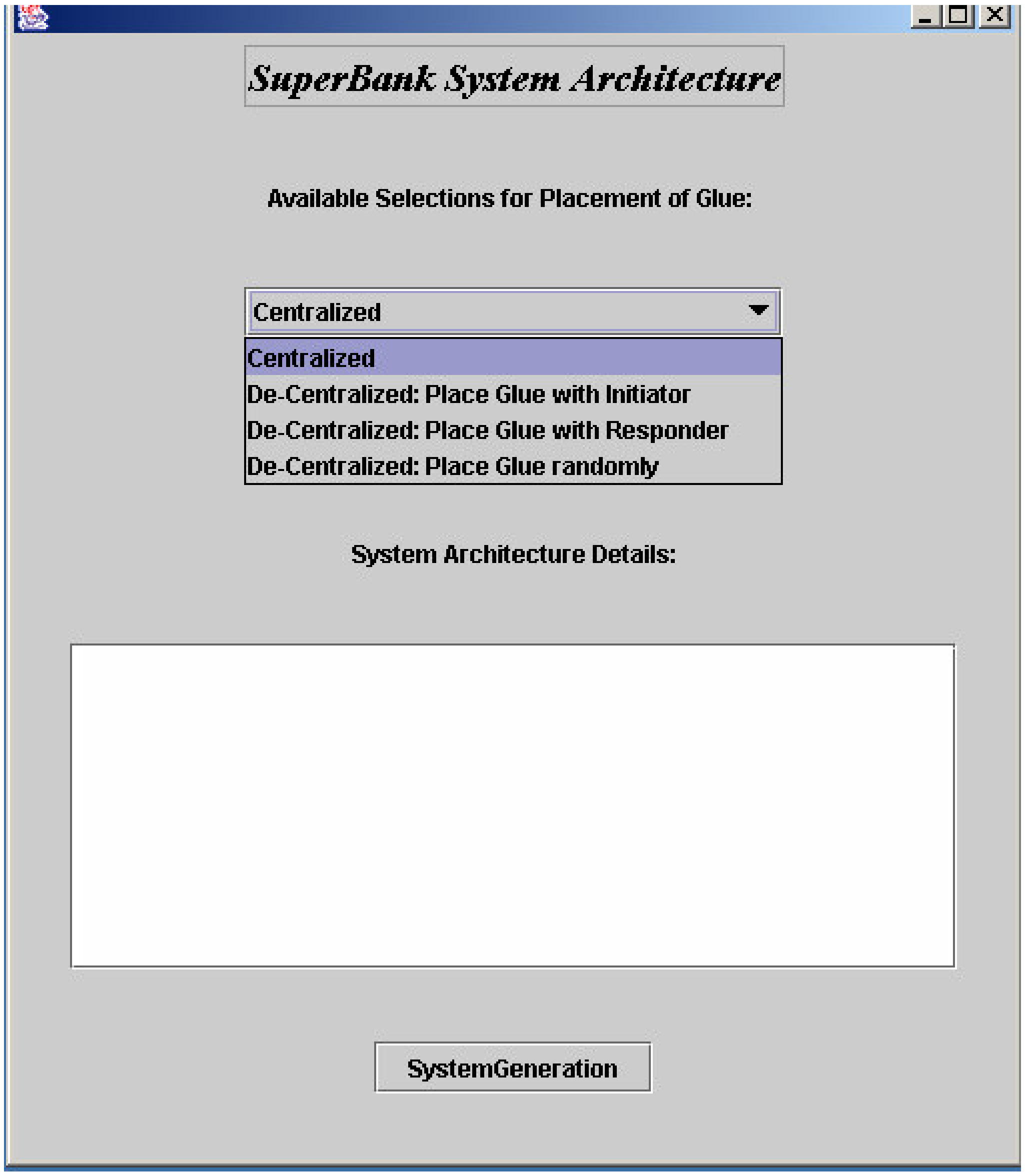
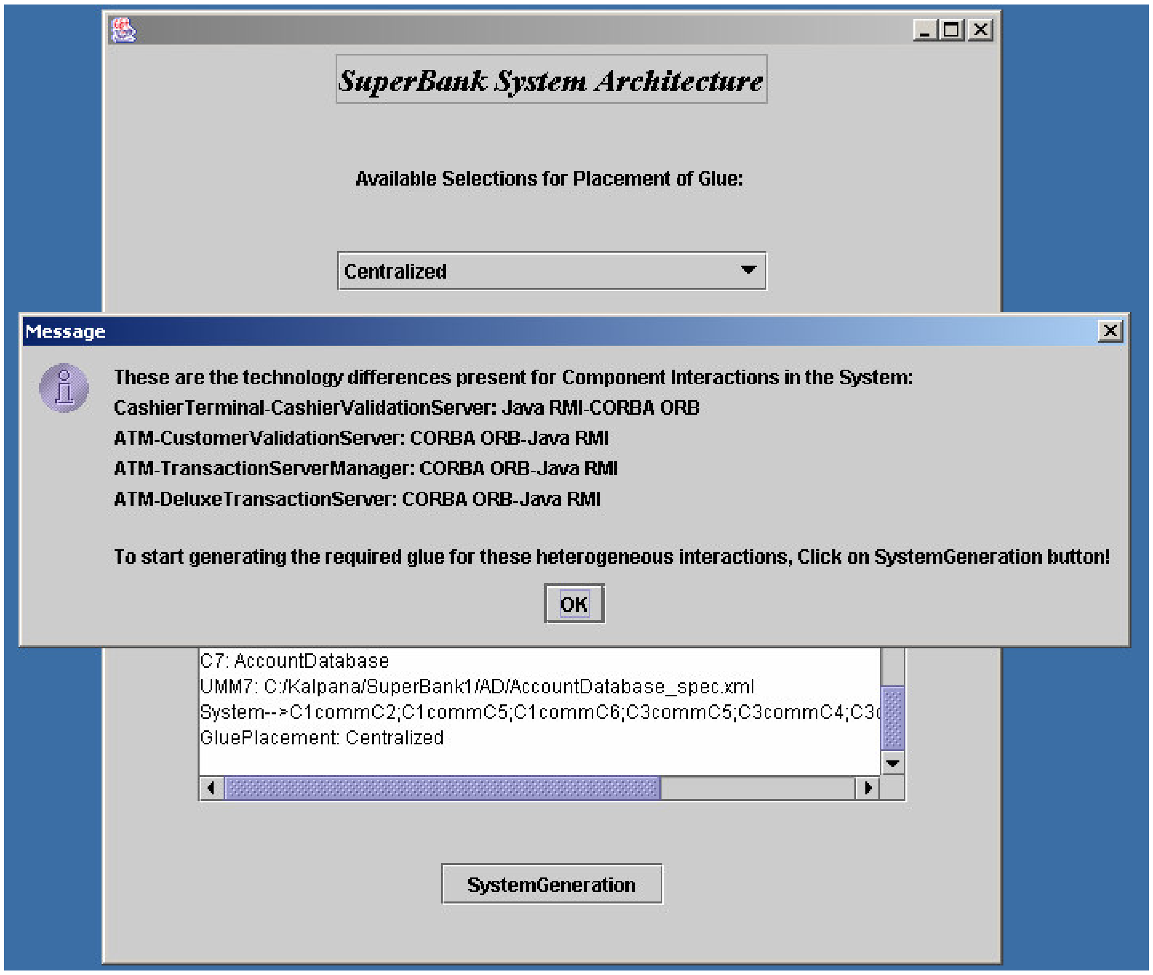

SuperBank System Architecture
Available Selections for Placement of Glue:
Centralized
Centralized
De-Centralized: Place Glue with Initiator
De-Centralized: Place Glue with Responder
De-Centralized: Place Glue randomly
System Architecture Details:
SystemGeneration

Figure 4.6 SysArchGUI view 1

Figure 4.7 shows the view after the “ShowSystemArchitecture” button is clicked with the “Centralized” placement option selected. It shows the 4 heterogeneous component interaction pairs in the form of “initiator component name – responder component name: initiator component technology – responder component technology”.

The technologies shown are Java RMI and CORBA, the implementation technologies of the components composing the "SuperBank" system.


SuperBank System Architecture
Available Selections for Placement of Glue:
Centralized
Message
OK
SystemGeneration

These are the technology differences present for Component Interactions in the System:
CashierTerminal-CashierValidationServer: Java RMI-CORBA ORB
ATM-CustomerValidationServer: CORBA ORB-Java RMI
ATM-TransactionServerManager: CORBA ORB-Java RMI
ATM-DeluxeTransactionServer: CORBA ORB-Java RMI

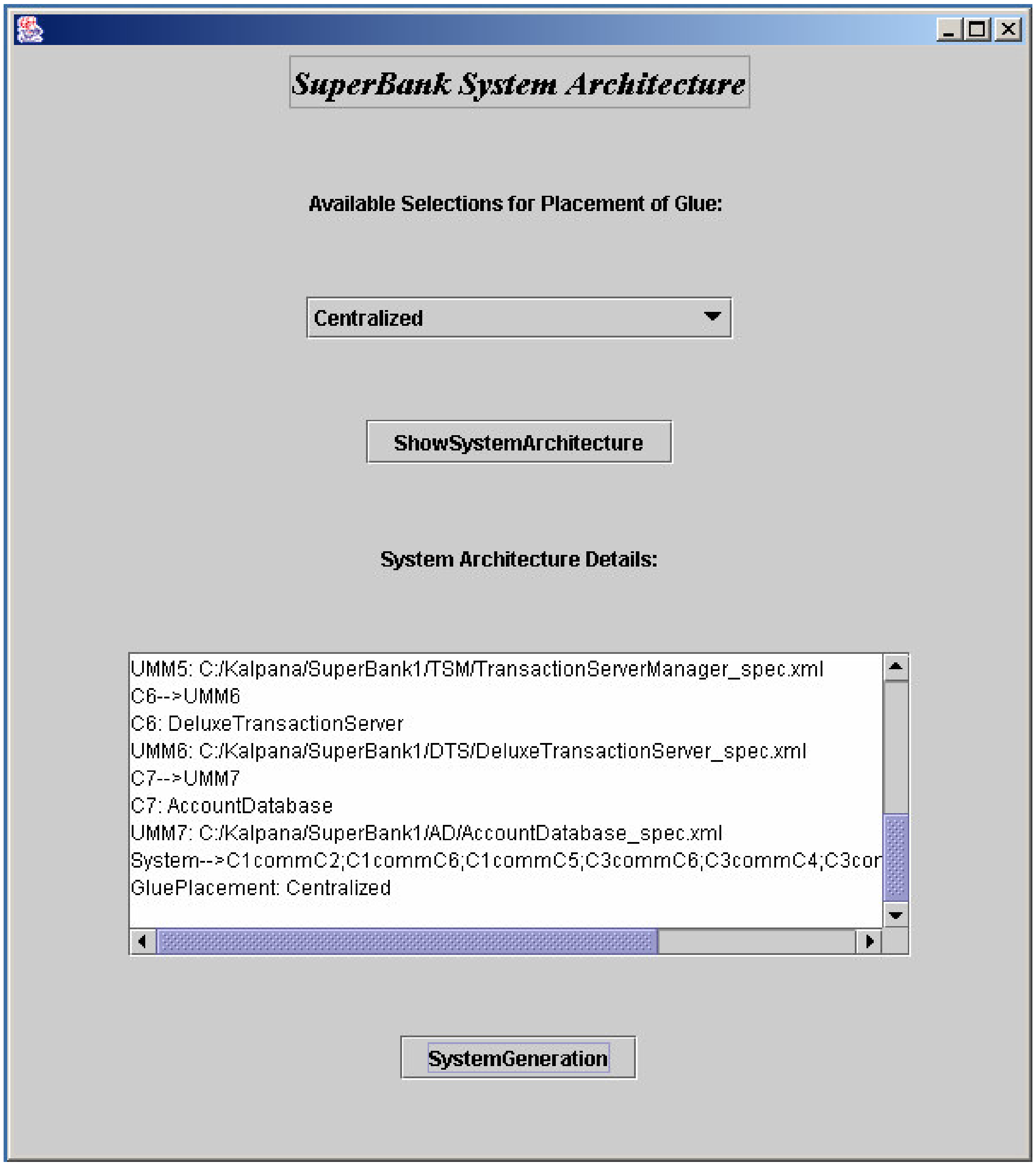
To start generating the required glue for these heterogeneous interactions, Click on SystemGeneration button!

```
C7: AccountDatabase
UMM7: C:/Kalpana/SuperBank1/AD/AccountDatabase_spec.xml
System-->C1commC2;C1commC5;C1commC6;C3commC5;C3commC4;C3
GluePlacement: Centralized
```

Figure 4.7 SysArchGUI view 2

Figure 4.8 shows the view after the "ok" button in the Figure 4.7 is clicked. It shows the architecture of the "SuperBank" system in a simple BNF notation.

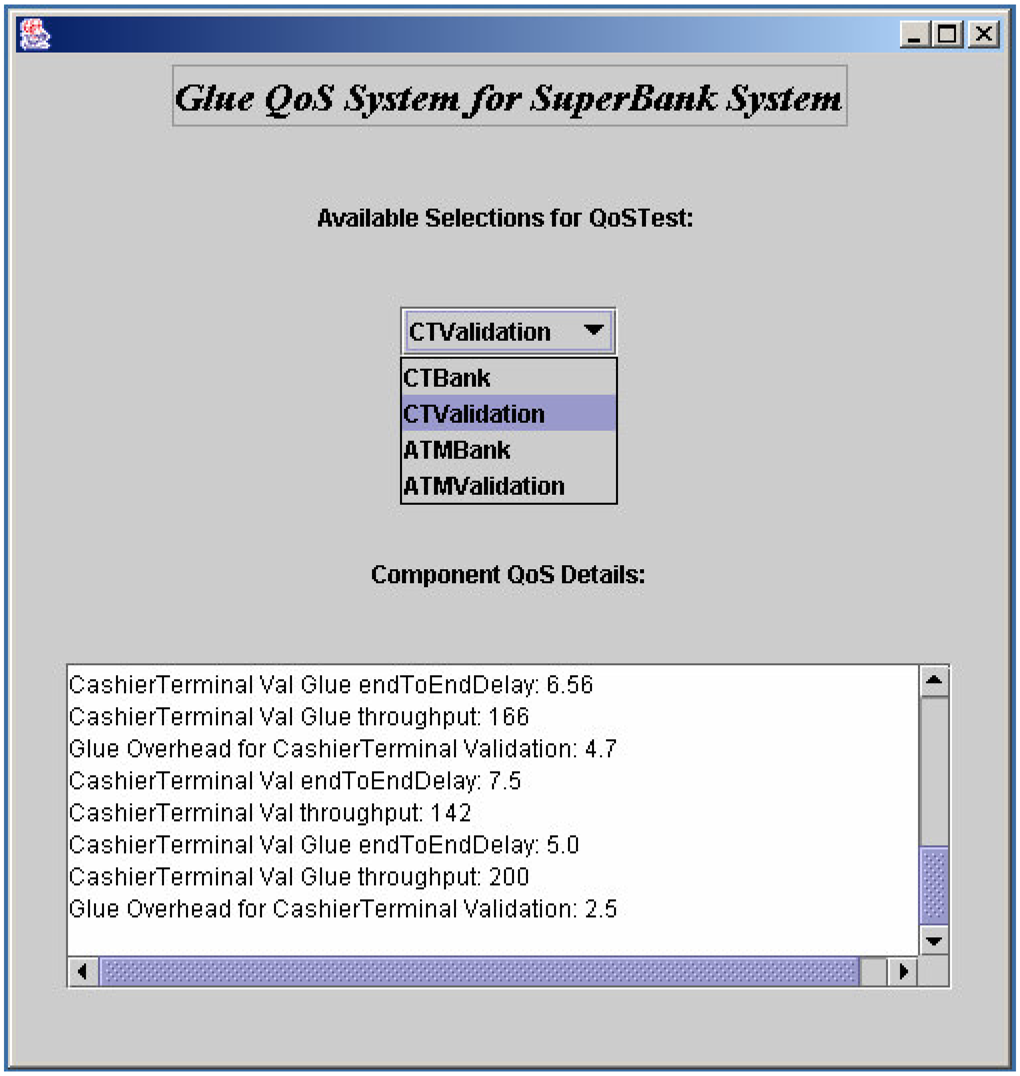
This BNF notation is designed for the purpose of showing the system architecture details. "C" followed by a number represents a component. When it is followed by a ":", it represents the name of the component but when it is followed by a "-->", it represents that the variable "UMM" followed by the same number represents the UMM specification URL of that particular component (in the form of "UMM" and ":"). When the variable "System" follows with an "-->", it represents all the interactions between components composing that system. The "comm" represents the communication pattern between the components. In case of "SuperBank", "comm" represents the request-reply protocol communication i.e., function calls. Figure 4.8 also shows the placement of glue option selected by the user through view 1 (Figure 4.6).


SuperBank System Architecture
Available Selections for Placement of Glue:
Centralized
ShowSystemArchitecture
System Architecture Details:
SystemGeneration

```
UMM5: C:/Kalpana/SuperBank1/TSM/TransactionServerManager_spec.xml
C6-->UMM6
C6: DeluxeTransactionServer
UMM6: C:/Kalpana/SuperBank1/DTS/DeluxeTransactionServer_spec.xml
C7-->UMM7
C7: AccountDatabase
UMM7: C:/Kalpana/SuperBank1/AD/AccountDatabase_spec.xml
System-->C1commC2;C1commC6;C1commC5;C3commC6;C3commC4;C3cor
GluePlacement: Centralized
```

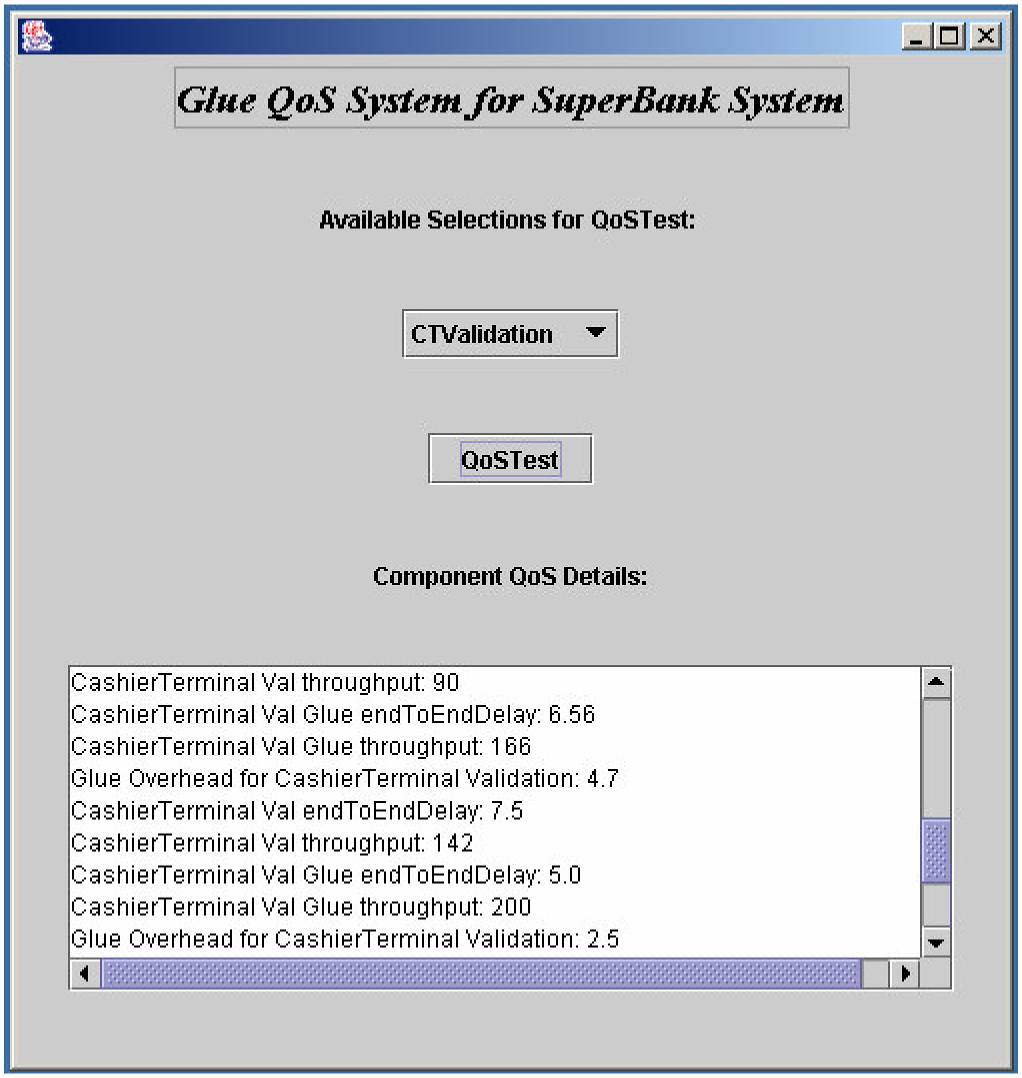
Figure 4.8 SysArchGUI view 3

GlueQoSGUI: After the glue is generated and deployed, GlueQoSGUI provides the System Integrator an ability to perform the QoS test on the System that has been composed, to obtain the overhead associated with the Glue components by clicking on the "QoSTest" button. Figures 4.9 & 4.10 illustrate the views of GlueQoSGUI. GlueQoSGUI is designed particularly for the "SuperBank" system. The four choices provided to the user through GlueQoSGUI are shown in Figure 4.9. These choices will be described in detail in the Chapter 5.



```
CashierTerminal Val Glue endToEndDelay: 6.56
CashierTerminal Val Glue throughput: 166
Glue Overhead for CashierTerminal Validation: 4.7
CashierTerminal Val endToEndDelay: 7.5
CashierTerminal Val throughput: 142
CashierTerminal Val Glue endToEndDelay: 5.0
CashierTerminal Val Glue throughput: 200
Glue Overhead for CashierTerminal Validation: 2.5
```

Figure 4.9 GlueQoSGUI view 1


Glue QoS System for SuperBank System
Available Selections for QoSTest:
CTValidation
QoSTest
Component QoS Details:

```
CashierTerminal Val throughput: 90
CashierTerminal Val Glue endToEndDelay: 6.56
CashierTerminal Val Glue throughput: 166
Glue Overhead for CashierTerminal Validation: 4.7
CashierTerminal Val endToEndDelay: 7.5
CashierTerminal Val throughput: 142
CashierTerminal Val Glue endToEndDelay: 5.0
CashierTerminal Val Glue throughput: 200
Glue Overhead for CashierTerminal Validation: 2.5
```

Figure 4.10 GlueQoSGUI view 2

Figure 4.10 shows the view after the “QoSTest” button is clicked by the user to obtain the experimental results which will be used in performance comparison study of various choices for placement of glue in Chapter 5.

This Chapter provided the prototype implementation details of the proposed UniGGen architecture. The next chapter provides the details of the experiments performed using this prototype to validate the UniGGen architecture proposed in chapter 3.

5. VALIDATION

In order to validate the performance of the prototype, an empirical experimentation was performed. The testing for the glue generation was performed using the prototype on a Banking system called BasicBank and SuperBank.

The service components comprising of BasicBank are as follows:

- ❖ CashierTerminal – a Java RMI component which requires services such as openAccount, closeAccount, depositMoney, withdrawMoney, transferMoney, checkBalance, validate.
- ❖ CashierValidationServer – a Java CORBA component which provides service to validate a CashierTerminal client.
- ❖ TransactionServerManager – a Java RMI component which provides services such as openAccount, closeAccount to a client.
- ❖ EconomicTransactionServer - a Java RMI component which provides services such as depositMoney, withdrawMoney, transferMoney, checkBalance to a client.

The service components comprising of SuperBank are as follows:

- ❖ CashierTerminal – a Java RMI component which requires services such as openAccount, closeAccount, depositMoney, withdrawMoney, transferMoney, checkBalance, validate.
- ❖ CashierValidationServer – a CORBA (Java or C++) component which provides service to validate a CashierTerminal client.
- ❖ ATM – a CORBA component which requires services such as depositMoney, withdrawMoney, transferMoney, checkBalance, validate.
- ❖ CustomerValidationServer – a Java RMI component which provides service to validate a ATM client.
- ❖ TransactionServerManager – a Java RMI component which provides services such as openAccount, closeAccount to a client.
- ❖ DeluxeTransactionServer - a Java RMI component which provides services such as depositMoney, withdrawMoney, transferMoney, checkBalance to a client.

❖ AccountDatabase – a Java RMI component acting as a database to store the transactions performed by DeluxeTransactionServer.

Experiments were carried out on three PCs running Windows 2000. Sun's JDK 1.4 was used to run the components of the UniGGen system. The components making up the banking system were deployed on these machines. All the RMI components comprising the banking system were registered with a single RMI Registry and all the CORBA components with a single CORBA Naming Service (for simplicity to depict the feature of Active Registry). The glue components generated and deployed were also registered at the same RMI Registry and CORBA Naming Service, i.e., the glue code generated for a CORBA to RMI call was registered in CORBA Naming Service and the glue code generated for a RMI to CORBA call was registered in RMI Registry.

The details, such as the system name, the component-UMM specification URL pairs of all the 7 components, and the 8 initiator-responder pairs, required pertaining to the SuperBank system were entered through the user interface. Then, one of the 4 choices for the placement of generated glue (as discussed in Section 3.4) was selected and the process of glue generation was started.

At present, the glue is placed automatically on centralized machine even though other choices are selected. To run the experiments for other choices, the glue was placed according to the choice on appropriate machines manually.

The experiments were at first carried out on a single machine i.e., the components comprising the UniGGen architecture and the generated glue components were all executed or deployed on the same machine. It caused a bottleneck as the load on the machine is more. The experiments were then carried out with 3 machines with the components comprising the banking system in a distributed manner (the CashierValidationServer on one, ATM on the second, and all the RMI components on the third machine), the components comprising the UniGGen on a single machine and the generated glue components were placed according to the choice selected for the placement.

5.1 Experimentations

The experiments were carried out for two different purposes:

1. To validate the proposed UniGGen architecture [discussed in Section 3.2]
2. To carry out a performance comparison between different choices proposed for the placement of the generated glue components [discussed in Section 3.4].

5.1.1 Experimentation to validate the proposed architecture

The experiments to validate the proposed UniGGen architecture were performed on three different banking systems to see whether the proposed architecture works for Java RMI and CORBA components:

1. The BasicBank system consisting of CashierValidationServer implemented in Java CORBA.
2. The SuperBank system consisting of CashierValidationServer implemented in Java CORBA.
3. The SuperBank system consisting of CashierValidationServer implemented in C++ CORBA.

The above three cases were chosen for experimentation because:

Case 1, if successful proves that the functionality is achieved for RMI and Java CORBA components for which the UniGGen architecture is proposed.

Case 2, if successful proves that the functionality is achieved for RMI and Java CORBA components and the proposed UniGGen architecture is believed to be scalable even though the experimentations were not carried out with many components.

Case 3, if successful proves that the functionality is achieved and that the proposed UniGGen is a generic architecture which can be applied to any RMI and CORBA components and not just to Java CORBA.

Hence, the above three cases can be used to state that the UniGGen architecture proposed serves its purpose and can be said that a template based approach is comprehensive enough to achieve interoperability between RMI and CORBA.

5.1.2 Performance comparison between different architectures for the placement of the generated glue

The measurements presented are averaged over 50 trials. The following performance metrics were gathered during the experimentation:

1. The Average Response Time (ART) in milliseconds, and is defined as the time taken to complete a request i.e., the duration between sending a request and receiving the result.
2. Length of the generated glue code in terms of number of lines.

The ART metric was selected for the following reason:

Since the call from an initiator to a responder component will pass through the glue component, it provides an ability to find out the overhead associated with the generated glue code on the performance of the system i.e., the extra time induced into the system composed due to the addition of the glue components. This is termed as glue overhead (in terms of ART), and is calculated by subtracting the ART obtained for the request to reply in between glue and responder component from the ART obtained for the request to reply in between initiator and responder component.

The second metric i.e., the length of the generated glue code was selected because it can help in finding out if the generated code changes depending on the topology or placement of the glue.

The Average Response Time was determined for the following cases using the SuperBank system consisting of a CashierValidationServer in Java CORBA:

- ❖ The SuperBank system without the requirement of glue code, i.e., the required glue code was already placed in the component by hand-crafting so that all the components can directly communicate with each other. This serves as a base case with which other cases are compared to.
- ❖ Placing and deploying the generated glue manually in a centralized manner, i.e., on a single machine.
- ❖ Placing and deploying the generated glue manually in a decentralized manner, i.e., on more than one machine
    - o Placing and deploying the generated glue required for the initiator components on the same machine as the initiator component requesting services.
    - o Placing and deploying the generated glue required for the initiator components on the same machine as the respective responder component providing services.
    - o Placing and deploying the generated glue required randomly on some machine. The random case was chosen such that its topology is different than any of the above two decentralized cases.

The above cases were selected to experiment with for the following reasons:

- ❖ The case without glue code as a separate component serves as the base case with which the performance results obtained from other cases can be compared to.
- ❖ The other cases are the choices proposed for the placement of the generated glue code and these will provide a complete insight to properly study the effect on performance of the system composed.

5.2 Results

The following sections provide the results obtained when the above experiments were carried out.

5.2.1 Results obtained from experiments carried out to validate the proposed UniGGen architecture

When the details of the System to be composed were entered through the user interface and provided to the prototype, the UniGGen system using pre-formulated templates generated glue code. The glue code was a compilable and deployable Java code that enabled proper communication between the components of the banking system. This proves that the UniGGen could compose the banking system in all the three cases, i.e., whether the number of components was 4 or 7, or whether the components of the SuperBank system were implemented in Java RMI and Java CORBA or C++ CORBA.

5.2.2 Performance comparison results

The performance comparison results between the choices for the placement of the generated glue were obtained for four types of requests made by the client-side components of the “SuperBank” system.

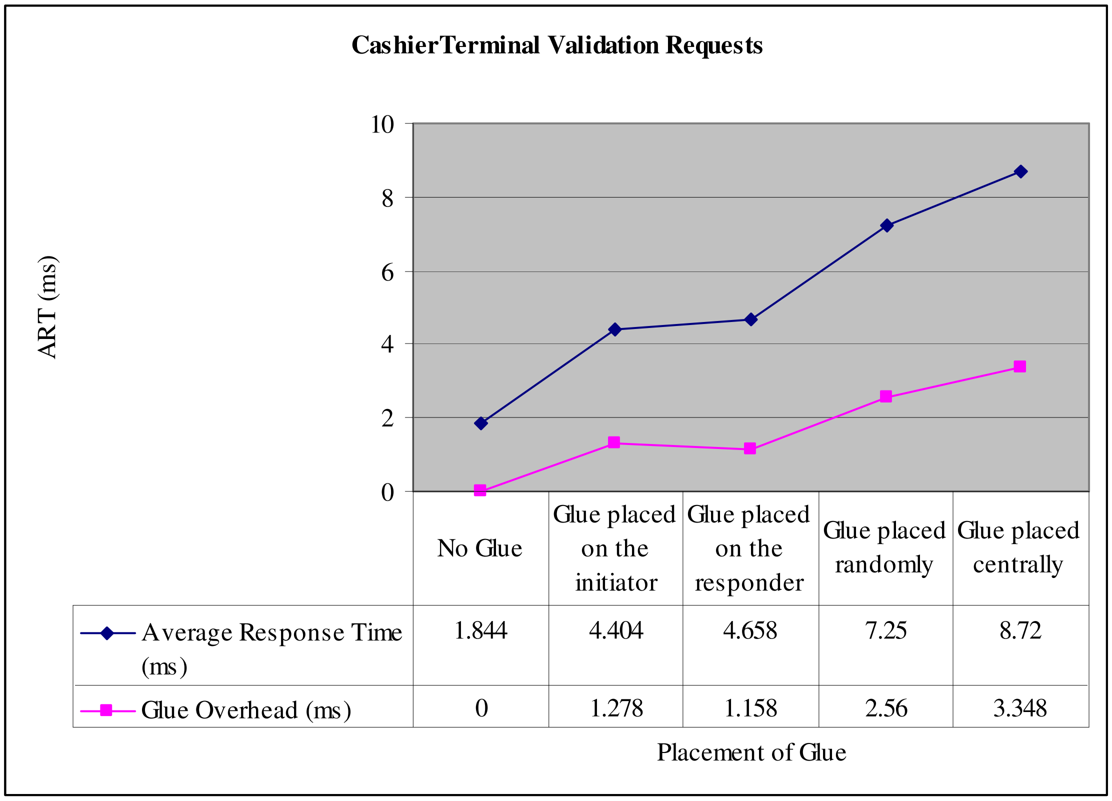
5.2.2.1 CashierTerminal validation request

Table 5.1 provides the performance comparison between different choices for the placement of the generated glue for validation requests made by CashierTerminal on CashierValidationServer. Figure 5.1 illustrates these results in the form of a graph.

The length of the glue code generated is 77 lines and does not vary for different placements of glue as the generated glue consists of only the code required for communication using request-reply mechanism between the components.

Placement of Glue	**Average Response Time (ms)**	**Glue Overhead (ms)**
No Glue	1.844	0
Glue placed on the initiator machine	4.404	1.278
Glue placed on the responder machine	4.658	1.158
Glue placed randomly	7.25	2.56
Glue placed centrally	8.72	3.348

Table 5.1 Performance comparison results for validation requests made by CashierTerminal



	No Glue	Glue placed on the initiator	Glue placed on the responder	Glue placed randomly	Glue placed centrally
◆ Average Response Time (ms)	1.844	4.404	4.658	7.25	8.72
■ Glue Overhead (ms)	0	1.278	1.158	2.56	3.348

Figure 5.1 Graph showing performance comparison results for validation requests made by CashierTerminal

The graph in Figure 5.1 shows that the ART and glue overhead are the lowest for the base case (i.e., without glue) and highest for the centralized placement of the glue. Of the other placement of glue choices, the glue when placed on the initiator machine or the responder machine gave approximately similar measurement for ART and glue overhead and is lesser compared to the random case. The base case is always better whereas, the source code of the components to be composed may not be accessible hence, there is a need for automatic generation of glue code.

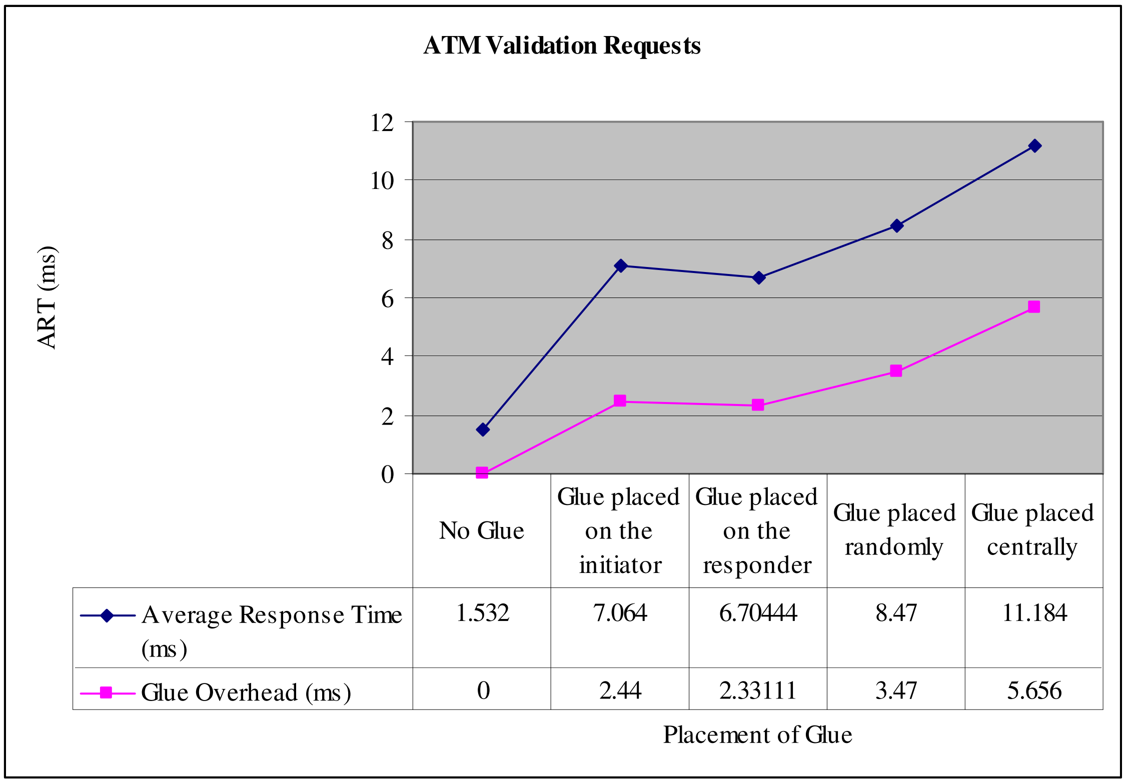
5.2.2.2 ATM validation request

Table 5.2 provides the performance comparison between different choices for the placement of the generated glue for validation requests made by an ATM on the CustomerValidationServer. Figure 5.2 illustrates these results in the form of a graph.

The length of the glue code generated is 85 lines and does not vary for different placements of glue as the generated glue consists of only the code required for communication using request-reply mechanism between the components.

Placement of Glue	**Average Response Time (ms)**	**Glue Overhead (ms)**
No Glue	1.532	0
Glue placed on the initiator machine	7.064	2.44
Glue placed on the responder machine	6.70444	2.33111
Glue placed randomly	8.47	3.47
Glue placed centrally	11.184	5.656

Table 5.2 Performance comparison results for validation requests made by ATM



	No Glue	Glue placed on the initiator	Glue placed on the responder	Glue placed randomly	Glue placed centrally
◆ Average Response Time (ms)	1.532	7.064	6.70444	8.47	11.184
■ Glue Overhead (ms)	0	2.44	2.33111	3.47	5.656

Figure 5.2 Graph showing performance comparison results for validation requests made by ATM

The graph in figure 5.2 shows that the ART and glue overhead are the lowest for the base case (i.e., without glue) and highest for the centralized placement of the glue. Of the other placement of glue choices, the glue when placed on the initiator machine or the responder machine gave approximately similar measurement for ART and glue overhead and is lesser compared to the random case. There is a sudden increase of ART and glue overhead from the case without glue to the placement of glue on the initiator machine because of the overload due to the UniGGen components running on the same machine where the CustomerValidationServer was running.

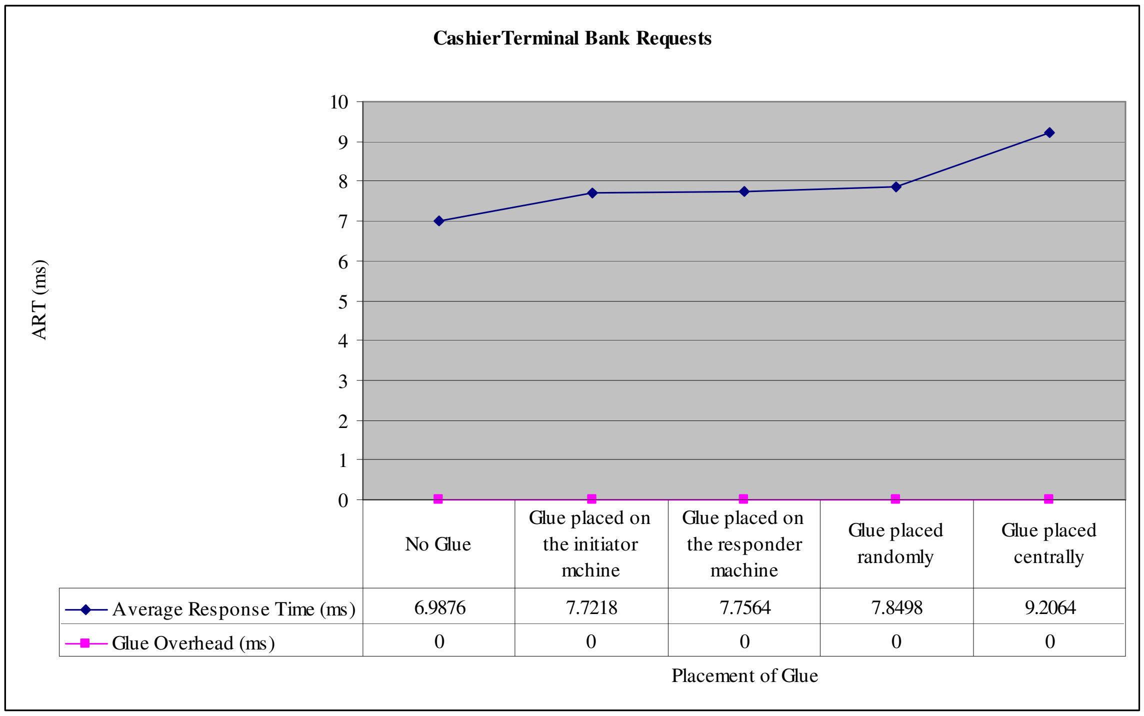
5.2.2.3 CashierTerminal bank request

Table 5.3 provides the performance comparison between different choices for the placement of the generated glue for bank transaction requests made by CashierTerminal on the TransactionServerManager and DeluxeTransactionServer components. The DeluxeTransactionServer in turn communicates with AccountDatabase. Figure 5.3 illustrates these results in the form of a graph.

The length of the glue code generated is 0 lines as glue is not required for these requests since the interactions required to complete these requests are not heterogeneous i.e., the CashierTerminal, TransactionServerManager, and DeluxeTransactionServer are all implemented using Java RMI.

Placement of Glue	**Average Response Time (ms)**	**Glue Overhead (ms)**
No Glue	6.9876	0
Glue placed on the initiator machine	7.7218	0
Glue placed on the responder machine	7.7564	0
Glue placed randomly	7.8498	0
Glue placed centrally	9.2064	0

Table 5.3 Performance comparison results for bank transaction requests made by CashierTerminal



	No Glue	Glue placed on the initiator mchine	Glue placed on the responder machine	Glue placed randomly	Glue placed centrally
◆ Average Response Time (ms)	6.9876	7.7218	7.7564	7.8498	9.2064
■ Glue Overhead (ms)	0	0	0	0	0

Figure 5.3 Graph showing performance comparison results for bank transaction requests made by CashierTerminal

The graph in figure 5.3 shows that the ART is the lowest for the base case (i.e., without glue) and highest for the centralized placement of the glue. The other placements of glue choices have approximately similar measurement for ART. The glue overhead is 0 in all the cases because there was no glue involved in the request and response and yet the results show a bottleneck with centralized placement.

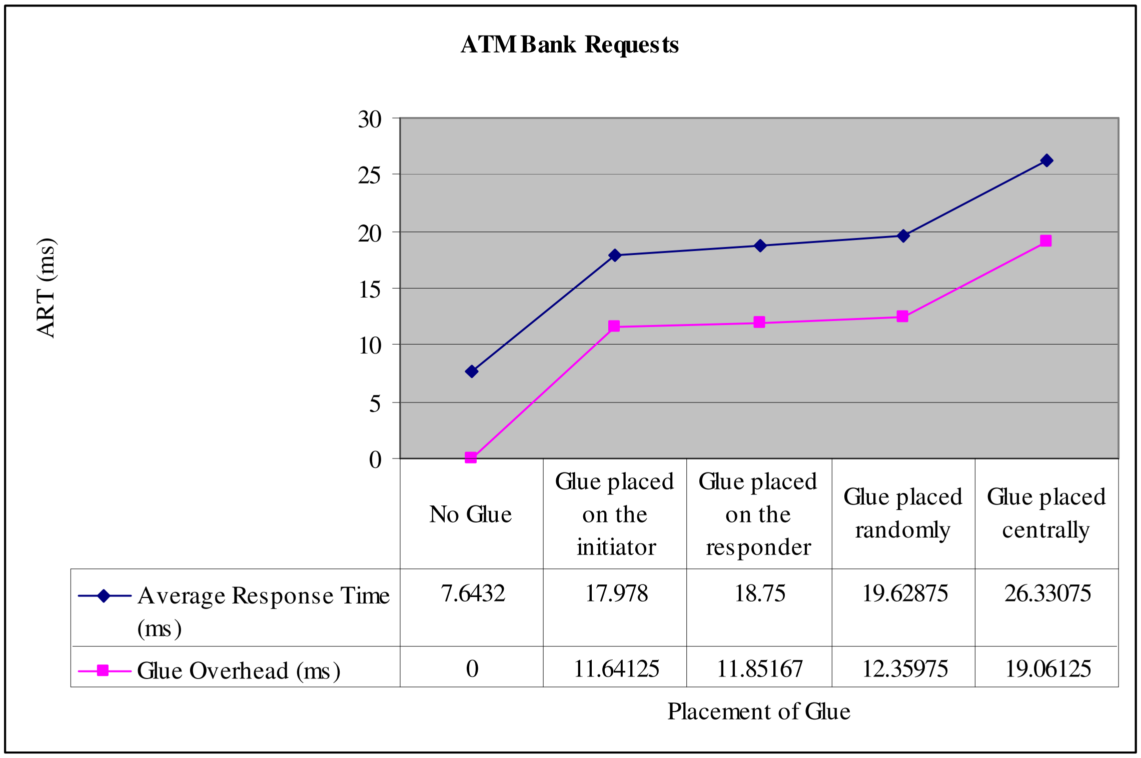
5.2.2.4 ATM bank request

Table 5.4 provides the performance comparison between different choices for the placement of the generated glue for bank transaction requests made by ATM on the TransactionServerManager and DeluxeTransactionServer components. The DeluxeTransactionServer in turn communicates with AccountDatabase. Figure 5.4 illustrates these results in the form of a graph.

The length of the glue code generated is 230 lines and does not vary for different placements of glue as the generated glue consists of only the code required for communication using request-reply mechanism between the components.

Placement of Glue	**Average Response Time (ms)**	**Glue Overhead (ms)**
No Glue	7.6432	0
Glue placed on the initiator machine	17.978	11.64125
Glue placed on the responder machine	18.75	11.85167
Glue placed randomly	19.62875	12.35975
Glue placed centrally	26.33075	19.06125

Table 5.4 Performance comparison results for bank transaction requests made by ATM



	No Glue	Glue placed on the initiator	Glue placed on the responder	Glue placed randomly	Glue placed centrally
◆ Average Response Time (ms)	7.6432	17.978	18.75	19.62875	26.33075
■ Glue Overhead (ms)	0	11.64125	11.85167	12.35975	19.06125

Figure 5.4 Graph showing performance comparison results for bank transaction requests made by ATM

The graph in figure 5.4 shows that the ART and glue overhead are the lowest for the base case (i.e., without glue) and highest for the centralized placement of the glue. Of the other placement of glue choices, the glue when placed on the initiator machine or the responder machine gave approximately similar measurement for ART and glue overhead and is lesser compared to the random case. The sudden increase of ART and glue overhead from the case without glue to the placement of glue on initiator machine is because of the large number of bank requests going through the glue components even though similar bank requests were performed in all cases.

The ART and glue overhead results obtained for bank requests consist of greater magnitude than that obtained for validation requests because the number and time taken to complete the requests in case of bank is more than that of validation.

The following Table 5.5 tabulates the results obtained for the length of the generated glue code for the various requests discussed above.

Interaction Request	**Length of the generated glue code (Number of lines)**
CashierTerminal validation request on CashierValidationServer	77
ATM validation request on CustomerValidationServer	85
CashierTerminal bank requests	0
ATM bank requests	230

Table 5.5 Results showing length of the generated glue code

The glue code required for validation of an ATM is more than that of CashierTerminal because the glue code for an ATM requires extra code for registering itself to a CORBA NamingService and to look up the CashierValidationServer. The glue code required for an ATM is more because it needs to contact two responder components i.e., TransactionServerManager and DeluxeTransactionServer. The generated glue code does not depend on the placement of the glue because, as for now, only a request-reply communication pattern is considered. It may vary if other communication patterns, such as message-passing, are considered.

The results obtained in all the cases of experimentation were as expected. Here are the conclusions drawn from the experiments carried out:

- ❖ The UniGGen architecture serves the functionality of achieving interoperability between Java RMI and CORBA technologies.
- ❖ The choices for the placement of the generated glue that place the glue on machines with either initiator or responder components are the better options as these add the least overhead on the performance of the system composed.

Hence, from the experiments conducted and results obtained, it can be concluded that the proposed UniGGen architecture is validated as the functionality is achieved and the system composed performs well with not much overhead compared to the situation without glue.

This chapter provided the experimentation details to validate the proposed UniGGen architecture. The next chapter provides the conclusion of this report and some future work.

6. CONCLUSION AND FUTURE WORK

This project presented the architecture and an implementation for a semi-automatic glue generation framework, the “Glue Generation Framework in UniFrame for CORBA-Java/RMI Interoperability”. The glue generation architecture itself forms a part of a framework, “UniFrame”, which aims at providing a platform for building DCS by integrating existing and emerging distributed component computing models under a common meta-model that enables discovery, interoperability, and collaboration of components via generative software techniques.

Once the UniGGen prototype was completed, testing was done to evaluate its functionality and also performance comparison between different choices proposed for the Placement of the generated glue code.

The following are the conclusions drawn from the experimental results shown in previous chapter:

- ❖ The results obtained indicate that the functionality, i.e., the glue generation for CORBA-Java/RMI interoperability is achieved by the proposed UniGGen architecture.
- ❖ The results obtained for the performance comparison between different choices for the placement of the generated glue indicate that the de-centralized choices which place and deploy the generated glue code on the machines with respective initiator and responder components provide better performance compared to other choices.

The contributions of the UniGGen project are:

- ❖ Provision of framework for semi-automatically generating the glue code required to interoperate between heterogeneous components implemented using Java RMI and CORBA object models using a template-based approach.
- ❖ Implementation of fully functional prototype for the UniGGen architecture.

- Study of the effect of the placement of the generated glue on the performance of the System composed.
- The architecture proposed achieves the interoperability between CORBA and Java RMI. Hence, templates can be used to solve the interoperability problem between components implemented in different underlying models.
- The process of designing templates requires manual intervention; however templates allow a more flexible approach to the generation of glue code with a higher degree of automation than is possible with a corresponding hand-crafted approach.
- The project suggests that the performance of the system composed depends on the placement of the glue and that the centralized placement causes bottleneck. The placement of glue with the respective initiator or responder components decreases the amount of overhead on the performance of the system composed.

The features of UniGGen are:

- The Glue code is generated automatically and the code generated supports exception-handling.
- The Glue code generated also contains the code for performing QoS testing (Turn-Around-Time) on the composed System.
- The proposed UniGGen architecture is believed to be scalable even though the experimentations were not carried out with many components.
- The UniGGen architecture supports bi-directional communication between Java RMI and CORBA components, i.e., a function call or a request-reply communication between a Java RMI component on a CORBA component and vice-versa.
- UniGGen provides flexibility by automating the process of glue code generation without modifying the source code (in-accessible) of the components to be composed.
- The automation performed by UniGGen requires less human effort in generating the glue, which means fewer errors.

Future work to complete for the UniGGen involves enhancing the implementation of the prototype and adding more UniGGen components to the prototype for further testing.

Some future work for the UniGGen and the prototype includes:

- This architecture can be extended to interoperate between other object models
  - Requires addition of pre-defined glue code templates for the respective object models into knowledgebase
  - May also require communication with bridges.
- This architecture can be extended to take care of other heterogeneity problems such as
  - Syntactic level
  - Semantic level
  - Protocol level
- Enhancing the existing knowledgebase by adding other systems to be composed.
  - At present, the systems that are present in knowledgebase are BasicBank and SuperBank only.
- Integration with the UniFrame System Generator.
  - The information present in the knowledgebase of UniGGen will be obtained from System Generator.
- Enhanced support for link failure detection.

In conclusion, this project has presented the Glue Generation Framework in UniFrame for the CORBA-Java/RMI Interoperability (UniGGen), which facilitates the expansion of the UniFrame Approach and extends the URDS. As the integration of separately and independently developed services becomes more prolific, a strong need exists for an effective and efficient method of generating the glue required for interoperation between these heterogeneous services. The UniGGen, coupled with the UniFrame Approach, represents a promising method for the discovery and integration of these services that are geographically scattered.

LIST OF REFERENCES

[1] Raje, R. R., "UMM: Unified Meta-object Model for Open Distributed Systems", Proceedings of ICA3PP 2000, 4th IEEE Int. Conf. Algorithms and Architecture for Parallel Processing", 2000, pp.454-465.

[2] Raje, R., Auguston, M., Bryant, B. R., Olson, A., Burt, C., "A Unified Approach for the Integration of Distributed Heterogeneous Software Components", Proceedings of the Monterey Workshop on Engineering Automation for Software Intensive System Integration, 2001, pp. 109-119.

[3] Raje, R., Auguston, M., Bryant, B. R., Olson, A., Burt, C., "A Quality of Service-based Framework for Creating Distributed Heterogeneous Software Components", Technical Report, Department of Computer and Information Science, Indiana University Purdue University Indianapolis, 2002.

[4] Orb2 Technical Overview, http://2ab.com/pdf/orb2_whitepaper.pdf.

[5] Batory, D., Geraci, B. J., "Validating Component Compositions in Software System Generators", Published in 1996 International Conference on Software Reuse, Orlando, Florida.

[6] Batory, D., Dasari, S., Geraci, B., Singhal, V., Sirkin, M., Thomas, J., "Achieving Reuse With Software System Generators", Published in IEEE Software, September 1995, pp. 89 - 94.

[7] The Web Services Community Portal, http://www.webServices.org, 2002.

[8] Mayo S., "Web Services: How Will Professional Services Firms Compete for This Multibillion-Dollar Opportunity", Proceedings of International Data Corporation (IDC), March 2002.

[9] Babel, Software tool, http://www.llnl.gov/CASC/components/babel.html.

[10] Alexandria, Software tool, http://www.llnl.gov/CASC/components/alexandria.html.

[11] Patel, M.I., Jordan, K., Clark, M., Bhatt, D., "Auto Source Code Generation and Run-Time Infrastructure and Environment for High Performance, Distributed Computing Systems", Proceedings of International Parallel (and Distributed) Processing Symposium (IPDPS) Workshops, 2000, pp. 816-822.

[12] Rodger, R.J., "Jostraca: a Template Engine for Generative Programming", Position Paper for the European Conference on Object-Oriented Programming Workshop on Generative Programming, 2002.

[13] Bulej, L., Bures, T., "A Connector model suitable for automatic generation of connectors", Technical Report, Department of Software Engineering, Charles University, Prague, 2003.

[14] Piccola, A Software Compositional Language, http://www.iam.unibe.ch/~scg/Research/Piccola/.

[15] Achermann, F., Nierstrasz, O., "Applications = Components + Scripts — A Tour of Piccola," Software Architectures and Component Technology, Mehmet Aksit (Ed.), pp. 261-292, Kluwer, 2001http://www.cs.ubc.ca/~gregor/cpsc511-00/resources/papers/tour-of-piccola.pdf.

[16] Java To Corba Bridge, Object Middleware Experts, http://www.omex.ch/downloads/JCorbaBridge_Info.pdf.

[17] CapeConnect Three Web Services Platform, Technical overview, http://www.capeclear.com/products/whitepapers/CapeConnectTwoforJ2EE.pdf.

[18] Raje, R., Auguston, M., Bryant, B. R., Olson, A., Burt, C., “A Unified Approach for the Integration of Distributed Heterogeneous Software Components”, Proceedings of the Monterey Workshop on Engineering Automation for Software Intensive System Integration, 2001, pp. 109-119.

[19] Nanditha N. Siram, "An Architecture for Discovery of Heterogeneous Software Components", MS Thesis, Department of Computer and Information Science, Indiana University Purdue University Indianapolis, March 2002.

[20] Raptis, K., Spinellis, D., Katsikas, S., “Distributed object bridges and Java-based object mediator” Published in Informatik / Informatique, 2:4-8, April 2000.

[21] Sun Microsystems, “JavaTM 2 Platform Enterprise Edition Specification, Version 1.3”, Sun Microsystems, August 2001, http://java.sun.com/j2ee/j2ee-1_3-fr-spec.pdf.

[22] Sun Microsystems, “Designing Enterprise Applications with the J2EETM Platform”, http://java.sun.com/blueprints/guidelines/designing_enterprise_applications/.

[23] Sun Microsystems, “Java 2 Platform, Standard Edition (J2SETM) v 1.4.0 Overview”, http://java.sun.com/j2se/1.4/.

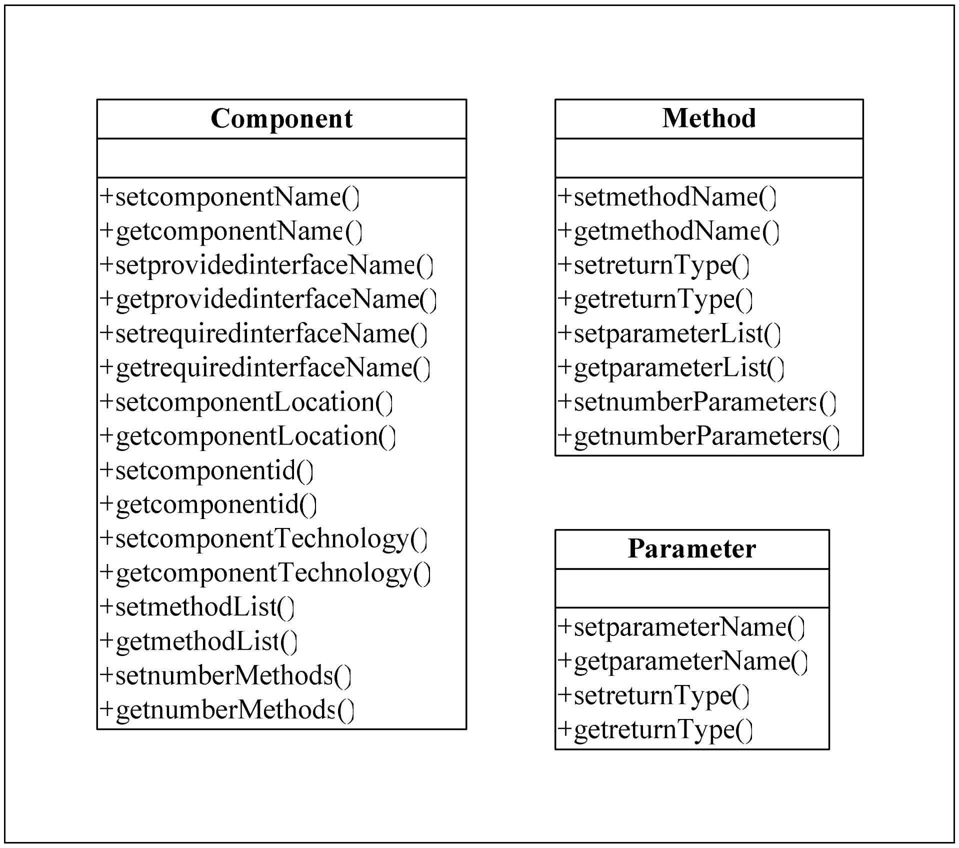
[24] Zhao, W., “Two-Level Grammar as the Formalism for Middleware Generation in Internet Component Broker Organization”, Proceedings of Generative and Component-Based Software Engineering/ Semantics, Applications, and Implementation of Program Generation (GCSE/SAIG) Young Researchers Workshop, held in conjunction with the First ACM SIGPLAN Conference on Generative Programming and Component Engineering, 2002.

[25] Zhao, W., “A Product Line Architecture for Component Model Domains”, Proceedings of PhD Students in Object Oriented Systems (PhDOOS) 2002, 12th

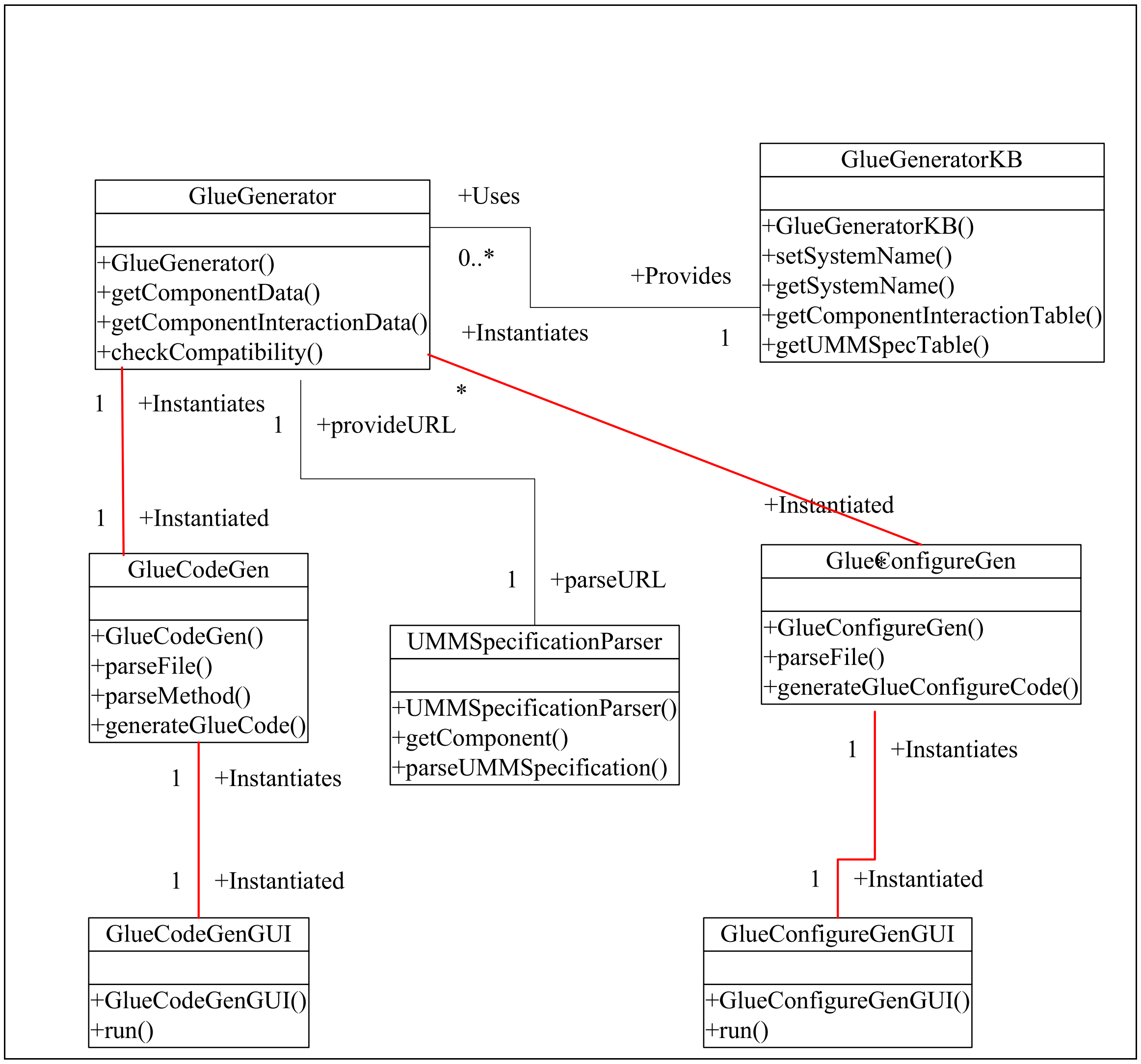
Workshop for PhD Students in Object-Oriented Systems, Malaga, Spain, June 2002.

APPENDICES

APPENDIX A: Class Diagrams for the Entity Objects



APPENDIX B: Class Diagrams for the Service Components



APPENDIX C: Source Code

Service Components

GlueGeneratorKB.java

```
/**
 * This class provides the knowledgebase required for the GlueGenerator System.
 *
 * @author Kalpana Tummala
 * @date April 2004
 * @version 1.0
 */

import java.io.*;
import java.util.*;

public class GlueGeneratorKB extends Thread implements Runnable{

		String SystemName;
		String ServerOption;
		String[][] ComponentInteractionTable;
		//ComponentInteractionTable: SystemName(0) - Inititaor(1) - Responder(2) pairs
		Hashtable ummSpecTable;
		int numberComponents = 0;
		int numberInteractions = 0;

		public GlueGeneratorKB() {
			ComponentInteractionTable = new String[8][3];

			ComponentInteractionTable[0][0] = "SuperBank";
			ComponentInteractionTable[0][1] = "CashierTerminal";
			ComponentInteractionTable[0][2] = "CashierValidationServer";

			ComponentInteractionTable[1][0] = "SuperBank";
			ComponentInteractionTable[1][1] = "CashierTerminal";
			ComponentInteractionTable[1][2] = "TransactionServerManager";

			ComponentInteractionTable[2][0] = "SuperBank";
			ComponentInteractionTable[2][1] = "CashierTerminal";
			ComponentInteractionTable[2][2] = "DeluxeTransactionServer";

			ComponentInteractionTable[3][0] = "SuperBank";
			ComponentInteractionTable[3][1] = "ATM";
			ComponentInteractionTable[3][2] = "CustomerValidationServer";

			ComponentInteractionTable[4][0] = "SuperBank";
			ComponentInteractionTable[4][1] = "ATM";
			ComponentInteractionTable[4][2] = "TransactionServerManager";

			ComponentInteractionTable[5][0] = "SuperBank";
			ComponentInteractionTable[5][1] = "ATM";
			ComponentInteractionTable[5][2] = "DeluxeTransactionServer";

			ComponentInteractionTable[6][0] = "SuperBank";
			ComponentInteractionTable[6][1] = "TransactionServerManager";
			ComponentInteractionTable[6][2] = "DeluxeTransactionServer";

			ComponentInteractionTable[7][0] = "SuperBank";
```

```
                    ComponentInteractionTable[7][1] = "DeluxeTransactionServer";
                    ComponentInteractionTable[7][2] = "AccountDatabase";

                    //ummSpecTable: Component name - ummSpecURL(xml)
                    ummSpecTable = new Hashtable();
                    ummSpecTable.put("CashierTerminal", C:/Kalpana/SuperBank1/CT/CashierTerminal_spec.xml");
                    ummSpecTable.put("CashierValidationServer",
C:/Kalpana/SuperBank1/CashVS/CashierValidationServer_spec.xml");
                    ummSpecTable.put("ATM", "C:/Kalpana/SuperBank1/ATM/ATM_spec.xml");
                    ummSpecTable.put("CustomerValidationServer",
"C:/Kalpana/SuperBank1/CustVS/CustomerValidationServer_spec.xml");
                    ummSpecTable.put("TransactionServerManager",
"C:/Kalpana/SuperBank1/TSM/TransactionServerManager_spec.xml");
                    ummSpecTable.put("EconomicTransactionServer",
"C:/Kalpana/SuperBank1/ETS/EconomicTransactionServer_spec.xml");
                    ummSpecTable.put("DeluxeTransactionServer",
"C:/Kalpana/SuperBank1/DTS/DeluxeTransactionServer_spec.xml");
                    ummSpecTable.put("AccountDatabase",
C:/Kalpana/SuperBank1/AD/AccountDatabase_spec.xml");
          }
          public String validSystemName(String SName){
                    String SysName = SName;
                    String validSysName = "";
                    if(SysName.equals("SuperBank")){
                              validSysName = "valid";
                    }
                    return validSysName;
          }
          public void setSystemName(String SystemName){
                    this.SystemName = SystemName;
                    if(SystemName.equals("SuperBank")){
                              numberComponents = 7;
                    }
                    if(SystemName.equals("SuperBank")){
                              numberInteractions = 8;
                    }
          }
          public String getSystemName(){
                    return SystemName;
          }
          public void setServerOption(String ServerOpt){
                    this.ServerOption = ServerOpt;
          }
          public String getServerOption(){
                    return ServerOption;
          }
          public int getnumberComponents(){
                    return numberComponents;
          }
          public int getnumberInteractions(){
                    return numberInteractions;
          }
          public String[][] getComponentInteractionTable(){
                    return ComponentInteractionTable;
          }
          public void setComponentInteractionTable(String[][] CITable){
                    this.ComponentInteractionTable = CITable;
          }
          public Hashtable getummSpecTable(){
                    return ummSpecTable;
          }
          public void setummSpecTable(Hashtable ummSTable){
```

```
                        this.ummSpecTable = ummSTable;
            }
}
```

GlueInterfaceGenerator.java

```
/**
 * This class provides the complete operation of the
 * interfaces generation required to compose the System.
 *
 * @author Kalpana Tummala
 * @date April 2004
 * @version 1.0
 */

import java.io.*;
import java.util.*;

public class GlueInterfaceGenerator extends Thread implements Runnable{

            static String SystemName;
            static String[][] ComponentInteractionTable;
            //ComponentInteractionTable: SystemName(0) - Inititaor(1) - Responder(2) pairs
            static Hashtable ummSpecTable;
            static int num;

            public GlueInterfaceGenerator(String SystemName) {
                        this.SystemName = SystemName;
            }

            public static void main(String args[]) {

            try{
                        GlueGeneratorKB GGKB = new GlueGeneratorKB();
                        ComponentInteractionTable = GGKB.getComponentInteractionTable();
                        ummSpecTable = GGKB.getummSpecTable();

                        GlueInterfaceGenerator glueGen = new GlueInterfaceGenerator(args[0]);
                        if(SystemName.equals("SuperBank")){
                                    num = 8;
                                    System.out.println("num = " + num);
                        }

                        String initiator = "";
                        String initiatorUMMspec = "";
                        String initiatorTech = "";
                        String responder = "";
                        String responderUMMspec = "";
                        String responderTech = "";
                        Runtime rt = Runtime.getRuntime();
                        String interfaceGen = "GlueInterfaceGen_CS_RC";
                        System.out.println("Initiator-Responder pairs: ");

                        for (int i = 0; i < num; i++){
                                    if(ComponentInteractionTable[i][0].equals(SystemName)){
                                                initiator = ComponentInteractionTable[i][1];
                                                initiatorUMMspec = (String)ummSpecTable.get(initiator);
                                                UMMSpecificationParser kpinitiator = new
UMMSpecificationParser(initiatorUMMspec);
                                                Component initiatorComp = new Component();
                                                initiatorComp = kpinitiator.getComponent();
                                                initiatorTech = initiatorComp.getcomponentTechnology();
```

```
                                        responder = ComponentInteractionTable[i][2];
                                        responderUMMspec = (String)ummSpecTable.get(responder);
                                        UMMSpecificationParser kpresponder = new
UMMSpecificationParser(responderUMMspec);
                                        Component responderComp = new Component();
                                        responderComp = kpresponder.getComponent();
                                        responderTech = responderComp.getcomponentTechnology();

                                        if(initiatorTech.equals("Java RMI") && responderTech.equals("CORBA
ORB")){
                                                  System.out.println("in the if for CS_RC");
                                                  System.out.println(SystemName + " : " + initiator + "/" + responder +
" " + initiatorTech + "/" + responderTech);

                                                  GlueInterfaceGen_CS_RC GIG_CS_RC = new
GlueInterfaceGen_CS_RC(responderComp);
                                                  GIG_CS_RC.start();
                                                  System.out.println("GlueInterfaceGen 1 done");

                            int x = System.in.read();
                                        }
                                        else if(initiatorTech.equals("CORBA ORB") && responderTech.equals("Java
RMI")){
                                                  System.out.println("in the if for RS_CC");
                                                  System.out.println(SystemName + " : " + initiator + "/" + responder +
" " + initiatorTech + "/" + responderTech);
                                                  GlueInterfaceGen_RS_CC GIG_RS_CC = new
GlueInterfaceGen_RS_CC(responderComp);
                                                  GIG_RS_CC.start();
                                                  System.out.println("GlueInterfaceGen 2 done");

                            int x = System.in.read();
                                        }
                              }
                    }
          }catch(Exception e){
                    System.out.println("GlueGenerator Exception: "+ e);
          }
          }
}
```

GlueInterfaceGen_CS_RC.java

```
/** This class is used to generate the interfaces that the glue code requires to
 *  interoperate between RMI and CORBA components using the templates
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class GlueInterfaceGen_CS_RC extends Thread implements Runnable{

          Component comp;
          String CSInterface_name;
          String method_name;
          String return_type;
```

```
		String parameters_signature;

		public GlueInterfaceGen_CS_RC(Component comp) {
				this.comp = comp;
		}

		public void run() {
				System.out.println("*******************************************************");
				CSInterface_name = comp.getprovidedinterfaceName();
				method_name = "";
				return_type = "";
				parameters_signature = "";
				parseFile();
				GlueInterfaceGenGUI_CS_RC GIGGUI = new
GlueInterfaceGenGUI_CS_RC(CSInterface_name);
				GIGGUI.start();
		}

		public void parseFile() {
				int filePosition;
				String tagContent;
				try {
						BufferedReader fileReader = new BufferedReader(new
FileReader("GlueInterface_CS_RC_Template.txt"));
						BufferedWriter fileWriter = new BufferedWriter(new FileWriter(CSInterface_name +
".java"));

						Hashtable returnMList = (Hashtable) comp.getmethodList();
						int nMethods = comp.getnumberMethods();
						System.out.println("********************************");
						System.out.println("Number of methods: " + nMethods);

						while( (filePosition=fileReader.read()) >= 0 ) {
								//System.out.println(filePosition);
								if (filePosition != '<'){
										fileWriter.write(filePosition);
								}else	{
										tagContent = "";
										while( (filePosition=fileReader.read()) != '>' ) {
												tagContent += (char)filePosition;
										}
										System.out.println(tagContent);
										// Compare the tagContents
										if ( tagContent.equals("CSInterface_name") ) {
												fileWriter.write(CSInterface_name);
										}else if ( tagContent.equals("METHODS") ) {
												for(int i=1;i <= nMethods;i++) {
														Integer mid = new Integer(i);
														Method m = (Method) returnMList.get(mid);
	System.out.println("********************************");
														method_name = m.getmethodName();
														return_type = m.getreturnType();
														if (return_type.equals("string")){
																return_type = "String";
														} else if (return_type.equals("long")){
																return_type = "int";
														}
														System.out.println("Method name: " +
method_name + " return type: " + return_type);
														Hashtable returnPList = (Hashtable)
m.getparameterList();
														int nParams = m.getnumberParameters();
														System.out.println("Number of params: " +
nParams);
														for(int j = 1;j <= nParams;j++) {
															Integer pid = new Integer(j);
															Parameter p = (Parameter)
returnPList.get(pid);
														System.out.println("*****************");
															String param_name =
p.getparameterName();
															String param_returntype =
p.getreturnType();
															if (param_returntype.equals("string")){
																param_returntype = "String";
															} else if
(param_returntype.equals("long")){
																param_returntype = "int";
															}
															System.out.println("Parameter name: "
+ param_name + " return type: " + param_returntype);
															if(j == nParams) {
																parameters_signature += " "
+ param_returntype + " " + param_name;
															}else {
																parameters_signature += " "
+ param_returntype + " " + param_name + ",";
															}
														}
														System.out.println("parameters_signature: " +
parameters_signature);
														parseMethod(fileWriter);
													}
												}
											}//end of else
										} // end of WHILE loop
										fileWriter.close();
									}catch(Exception e){
										System.out.println("Parse File Exception: " + e);
										e.printStackTrace();
									}
		}//end pf parseFile

		public void parseMethod(BufferedWriter fileWriter) {
			int mfilePosition;
			String mtagContent;
			try {
				BufferedReader mfileReader = new BufferedReader(new
FileReader("GlueInterfaceMethods_CS_RC_Template.txt"));
				while( (mfilePosition=mfileReader.read()) >= 0 ) {
					//System.out.println(filePosition);
					if (mfilePosition != '<'){
						fileWriter.write(mfilePosition);
					}else	{
						mtagContent = "";
						while( (mfilePosition=mfileReader.read()) != '>' ) {
							mtagContent += (char)mfilePosition;
						}
						System.out.println(mtagContent);
						// Compare the tagContents
						if ( mtagContent.equals("method_name") ) {
							fileWriter.write(method_name);

						}else if ( mtagContent.equals("return_type") ) {
```

```
                                                        fileWriter.write(return_type);
                                                }else if ( mtagContent.equals("parameters_signature") ) {
                                                        fileWriter.write(parameters_signature);
                                                }
                                        }//end of else
                                } // end of WHILE loop
                                parameters_signature = "";
                        }catch(Exception e){
                                System.out.println("Parse Method Exception: " + e);
                                e.printStackTrace();
                        }
                }//end pf parseMethod
}//end of GlueInterfaceGen_CS_RC class
```

GlueInterfaceGenGUI_CS_RC.java

```
/** This class is used to compile and deploy the generated interfaces
 *  required to interoperate between RMI and CORBA components
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.util.*;
import java.io.*;

public class GlueInterfaceGenGUI_CS_RC extends Thread implements Runnable{
        String interfaceName;
        public GlueInterfaceGenGUI_CS_RC(String interfaceName){
                this.interfaceName = interfaceName;
        }
        public void run() {
                try {
                        Runtime rt = Runtime.getRuntime();
                        // To compile GlueInterface
                        Process proc3 = rt.exec("cmd /c start javac " + interfaceName + ".java");
                        InputStreamReader isr3 = new InputStreamReader(proc3.getInputStream());
                        BufferedReader br3 = new BufferedReader(isr3);
                        String line3 = null;
                        while ( (line3 = br3.readLine()) != null)
                                System.out.println(line3);
                        int exitVal3 = proc3.waitFor();
                        if(exitVal3==0)
                                System.out.println("GlueInterfaceGenGUI_CS_RC File: "+ interfaceName +
".java Successfully Compiled. ");
                        else
                                System.out.println("GlueInterfaceGenGUI_CS_RC File: "+ interfaceName +
".java Compiled with errors. ");
                } catch (Throwable t) {
                        t.printStackTrace();
                }
        }
}
```

GlueInterfaceGen_RS_CC.java

```
/** This class is used to generate the interfaces that the glue code requires to
 *  interoperate between CORBA and RMI components using the templates
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class GlueInterfaceGen_RS_CC extends Thread implements Runnable{

        Component comp;
        String RSInterface_name;
        String method_name;
        String return_type;
        String parameters_signature;

        public GlueInterfaceGen_RS_CC(Component comp) {
                super();
                this.comp = comp;
        }

        public void run() {
                //SETTING VALUES
                RSInterface_name = comp.getprovidedinterfaceName();
                System.out.println("RSInterface_name: " + RSInterface_name);
                System.out.println("**********************************************************");
                method_name = "";
                return_type = "";
                parameters_signature = "";
                parseFile();
                GlueInterfaceGenGUI_RS_CC GIGGUI = new
GlueInterfaceGenGUI_RS_CC(RSInterface_name.substring(0,4));
                GIGGUI.start();
        }

        public void parseFile() {
                int filePosition;
                String tagContent;
                try {
                        BufferedReader fileReader = new BufferedReader(new
FileReader("GlueInterface_RS_CC_Template.txt"));
                        BufferedWriter fileWriter = new BufferedWriter(new
FileWriter(RSInterface_name.substring(0,4) + ".idl"));

                        Hashtable returnMList = (Hashtable) comp.getmethodList();
                        int nMethods = comp.getnumberMethods();
                        System.out.println("******************************");
                        System.out.println("Number of methods: " + nMethods);

                        while( (filePosition=fileReader.read()) >= 0 ) {
                                //System.out.println(filePosition);
                                if (filePosition != '<'){
                                        fileWriter.write(filePosition);
                                }else   {
                                        tagContent = "";
                                        while( (filePosition=fileReader.read()) != '>' ) {
                                                tagContent += (char)filePosition;
                                        }
                                        System.out.println(tagContent);
                                        // Compare the tagContents
                                        if ( tagContent.equals("RSInterface_name") ) {
                                                fileWriter.write(RSInterface_name);
                                        }else if ( tagContent.equals("METHODS") ) {
                                                for(int i=1;i <= nMethods;i++) {
                                                        Integer mid = new Integer(i);
                                                        Method m = (Method) returnMList.get(mid);

        System.out.println("******************************");
                                                        method_name = m.getmethodName();
                                                        return_type = m.getreturnType();
                                                        if (return_type.equals("String")){
                                                                return_type = "string";
                                                        } else if (return_type.equals("int")){
                                                                return_type = "long";
                                                        }
                                                        System.out.println("Method name: " +
method_name + " return type: " + return_type);

                                                        Hashtable returnPList = (Hashtable)
m.getparameterList();
                                                        int nParams = m.getnumberParameters();
                                                        System.out.println("Number of params: " +
nParams);
                                                        for(int j = 1;j <= nParams;j++) {
                                                                Integer pid = new Integer(j);
                                                                Parameter p = (Parameter)
returnPList.get(pid);

        System.out.println("******************");
                                                                String param_name =
p.getparameterName();
                                                                String param_returntype =
p.getreturnType();
                                                                if (param_returntype.equals("String")){
                                                                        param_returntype = "string";
                                                                } else if
(param_returntype.equals("int")){

                                                                        param_returntype = "long";
                                                                }
                                                                System.out.println("Parameter name: "
+ param_name + " return type: " + param_returntype);
                                                                if(j == nParams) {
                                                                        parameters_signature += " "
+ "in" + " " + param_returntype + " " + param_name;
                                                                }else {
                                                                        parameters_signature += " "
+ "in" + " " + param_returntype + " " + param_name + ",";
                                                                }

                                                        }
                                                        System.out.println("parameters_signature: " +
parameters_signature);
                                                        parseMethod(fileWriter);
                                                }
                                        }
                                }//end of else
                        } // end of WHILE loop
                        fileWriter.close();
                }catch(Exception e){
                        System.out.println("Parse File Exception: " + e);
                        e.printStackTrace();
                }
        }//end pf parseFile

        public void parseMethod(BufferedWriter fileWriter) {
```

```
			int mfilePosition;
			String mtagContent;
			try {
				BufferedReader mfileReader = new BufferedReader(new
FileReader("GlueInterfaceMethods_RS_CC_Template.txt"));
				while( (mfilePosition=mfileReader.read()) >= 0 ) {
					//System.out.println(filePosition);
					if (mfilePosition != '<'){
						fileWriter.write(mfilePosition);
					}else	{
						mtagContent = "";
						while( (mfilePosition=mfileReader.read()) != '>' ) {
							mtagContent += (char)mfilePosition;
						}
						System.out.println(mtagContent);
						// Compare the tagContents
						if ( mtagContent.equals("method_name") ) {
							fileWriter.write(method_name);

						}else if ( mtagContent.equals("return_type") ) {
							fileWriter.write(return_type);
						}else if ( mtagContent.equals("parameters_signature") ) {
							fileWriter.write(parameters_signature);
						}
					}//end of else
				} // end of WHILE loop
				parameters_signature = "";
			}catch(Exception e){
				System.out.println("Parse Method Exception: " + e);
				e.printStackTrace();
			}
		}//end pf parseMethod

}//end of GlueInterfaceGen_RS_CC class
```

GlueInterfaceGenGUI_RS_CC.java

```
/** This class is used to generate the interfaces required to
 *  interoperate between CORBA and RMI components using the templates
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.util.*;
import java.io.*;

public class GlueInterfaceGenGUI_RS_CC extends Thread implements Runnable {
		String interfaceName;
		public GlueInterfaceGenGUI_RS_CC(String interfaceName){
			super();
			this.interfaceName = interfaceName;
		}

		public void run() {
			try {
				Runtime rt = Runtime.getRuntime();

				// To compile GlueInterface
				Process proc3 = rt.exec("cmd /c start idlc -all -d ./ " + interfaceName + ".idl");
				InputStreamReader isr3 = new InputStreamReader(proc3.getInputStream());
```

```
                                        BufferedReader br3 = new BufferedReader(isr3);
                                        String line3 = null;
                                        while ( (line3 = br3.readLine()) != null)
                                                System.out.println(line3);
                                        int exitVal3 = proc3.waitFor();
                                        if(exitVal3==0)
                                                System.out.println("GlueInterfaceGenGUI_RS_CC File: "+ interfaceName +
".idl Successfully Compiled. ");
                                        else
                                                System.out.println("GlueInterfaceGenGUI_RS_CC File: "+ interfaceName +
".idl Compiled with errors. ");

        int x = System.in.read();

                                        Process proc2 = rt.exec("cmd /c start javac -classpath
%ORB2%/lib/orb2.jar;C:/Kalpana/xmlParser/xerces.jar;. " + interfaceName + "*.java _" + interfaceName + "*.java");
                                        InputStreamReader isr2 = new InputStreamReader(proc2.getInputStream());
                                        BufferedReader br2 = new BufferedReader(isr2);
                                        String line2 = null;
                                        while ( (line2 = br2.readLine()) != null)
                                                System.out.println(line2);
                                        int exitVal2 = proc2.waitFor();
                                        System.out.println("GlueCodeGenGUI_RS_CC " + interfaceName + " exitVal2 " +
exitVal2);
                                        if(exitVal2==0)
                                                System.out.println("GlueCodeGenGUI_RS_CC All Files: Successfully
Compiled. ");
                                        else
                                                System.out.println("GlueCodeGenGUI_RS_CC Compiled with errors. ");

                                int z = System.in.read();

                                } catch (Throwable t) {
                                        t.printStackTrace();
                                }
                }
}
```

GlueGenerator.java

```
/**
 * This class provides the complete operation of the Glue Generation System
 * starting from obtaining the information required from the KB and then
 * generating the glue code using GlueCodeGen, etc and finally deploying it.
 *
 * @author Kalpana Tummala
 * @date April 2004
 * @version 1.0
 */

import java.io.*;
import java.util.*;

public class GlueGenerator extends Thread implements Runnable{

        String SystemName;
        String ServerOption;
        String[][] ComponentInteractionTable;
        //ComponentInteractionTable: SystemName(0) - Inititaor(1) - Responder(2) pairs
        Hashtable ummSpecTable;
        int numberInteractions;
        GlueGeneratorKB GlueGeneratorKB;
	public GlueGenerator(GlueGeneratorKB GlueGeneratorKB) {
		this.GlueGeneratorKB = GlueGeneratorKB;
	}

	public void startGG() {
		try{
		ComponentInteractionTable = GlueGeneratorKB.getComponentInteractionTable();
		ummSpecTable = GlueGeneratorKB.getummSpecTable();

		SystemName = GlueGeneratorKB.getSystemName();
		ServerOption = GlueGeneratorKB.getServerOption();
		System.out.println("SysName: " + SystemName + "ServerOption: " + ServerOption);

		numberInteractions = GlueGeneratorKB.getnumberInteractions();

		String initiator = "";
		String initiatorUMMspec = "";
		String initiatorTech = "";
		String responder = "";
		String responderUMMspec = "";
		String responderTech = "";
		String init_resp_pair = "";
		Runtime rt = Runtime.getRuntime();
		String interfaceGen = "GlueInterfaceGen_CS_RC";

		System.out.println("Initiator-Responder pairs: ");

		for (int i = 0; i < numberInteractions; i++){
			if(ComponentInteractionTable[i][0].equals(SystemName)){
				initiator = ComponentInteractionTable[i][1];
				initiatorUMMspec = (String)ummSpecTable.get(initiator);
				UMMSpecificationParser kpinitiator = new
UMMSpecificationParser(initiatorUMMspec);
				Component initiatorComp = new Component();
				initiatorComp = kpinitiator.getComponent();
				initiatorTech = initiatorComp.getcomponentTechnology();

				responder = ComponentInteractionTable[i][2];
				}
				responderUMMspec = (String)ummSpecTable.get(responder);
				UMMSpecificationParser kpresponder = new
UMMSpecificationParser(responderUMMspec);
				Component responderComp = new Component();
				responderComp = kpresponder.getComponent();
				responderTech = responderComp.getcomponentTechnology();

				init_resp_pair = initiator + "-->" + responder;

				if(initiatorTech.equals("Java RMI") && responderTech.equals("CORBA
ORB")){
					System.out.println("in the if for CS_RC");
					System.out.println(SystemName + " : " + initiator + "/" + responder +
" " + initiatorTech + "/" + responderTech);

					GlueCodeGen_CS_RC GCG_CS_RC = new
GlueCodeGen_CS_RC(responderComp, init_resp_pair);
					GCG_CS_RC.start();
					System.out.println("GlueCodeGen 1 started");

       int y = System.in.read();
```

```
                                                GlueConfigureGen_CS_RC GCfG_CS_RC = new
GlueConfigureGen_CS_RC(initiatorComp, responderComp, init_resp_pair);
                                                GCfG_CS_RC.start();
                                                System.out.println("GlueConfigureGen 1 started");

        int z = System.in.read();

                                        }
                                        else if(initiatorTech.equals("CORBA ORB") && responderTech.equals("Java
RMI")){
                                                System.out.println("in the if for RS_CC");
                                                System.out.println(SystemName + " : " + initiator + "/" + responder +
" " + initiatorTech + "/" + responderTech);

                                                GlueCodeGen_RS_CC GCG_RS_CC = new
GlueCodeGen_RS_CC(responderComp, init_resp_pair);
                                                GCG_RS_CC.start();
                                                System.out.println("GlueCodeGen 2 started");

        int y = System.in.read();

                                                GlueConfigureGen_RS_CC GCfG_RS_CC = new
GlueConfigureGen_RS_CC(initiatorComp, responderComp, init_resp_pair);
                                                GCfG_RS_CC.start();
                                                System.out.println("GlueConfigureGen 2 started");

        int z = System.in.read();

                                        }
                                }
}
          }catch(Exception e){
                    System.out.println("GlueGenerator Exception: "+ e);
          }
          }
}
```

GlueCodeGen_CS_RC.java

```
/** This class is used to generate the glue code required to
 *  interoperate between RMI and CORBA components using the templates
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class GlueCodeGen_CS_RC extends Thread implements Runnable{

          Component comp;
          String CSInterface_name;
          String method_name;
          String return_type;
          String parameters_signature;
          String parameters_name;
          String GLUECompName;
          String init_resp_pair;
	public GlueCodeGen_CS_RC(Component comp, String init_resp_pair) {
		super();
		this.comp = comp;
		this.init_resp_pair = init_resp_pair;
	}

	public void run() {

		//GETTING & SETTING VALUES
		System.out.println("*********************************************************");
		CSInterface_name = comp.getprovidedinterfaceName();
		GLUECompName = "Glue_CS_RC_" + CSInterface_name.substring(0,4);

		method_name = "";
		return_type = "";
		parameters_signature = "";
		parameters_name = "";
		parseFile();
		GlueCodeGenGUI_CS_RC GCGGUI = new GlueCodeGenGUI_CS_RC(GLUECompName);
		GCGGUI.start();
	}

	// parses the template and writes into Glue code file
	public void parseFile() {
		int filePosition;
		String tagContent;
		try {
			BufferedReader fileReader = new BufferedReader(new
FileReader("GlueCode_CS_RC_Template.txt"));
			BufferedWriter fileWriter = new BufferedWriter(new FileWriter(GLUECompName +
".java"));

			Hashtable returnMList = (Hashtable) comp.getmethodList();
			int nMethods = comp.getnumberMethods();
			System.out.println("*******************************");
			System.out.println("Number of methods: " + nMethods);

			while( (filePosition=fileReader.read()) >= 0 ) {
				//System.out.println(filePosition);
				if (filePosition != '<'){
					fileWriter.write(filePosition);
				}else	{
					tagContent = "";
					while( (filePosition=fileReader.read()) != '>' ) {
						tagContent += (char)filePosition;
					}
					System.out.println(tagContent);
					// Compare the tagContents
					if ( tagContent.equals("Glue_CS_RC") ) {
						fileWriter.write(GLUECompName);
					}else if ( tagContent.equals("CSInterface_name") ) {
						fileWriter.write(CSInterface_name);
					}else if ( tagContent.equals("init_resp_pair") ) {
						fileWriter.write(init_resp_pair);
					}else if ( tagContent.equals("METHODS") ) {
						for(int i=1;i <= nMethods;i++) {
							Integer mid = new Integer(i);
							Method m = (Method) returnMList.get(mid);
					System.out.println("*******************************");
							method_name = m.getmethodName();
							return_type = m.getreturnType();
							if (return_type.equals("long")) {
```

```
                                                                                return_type = "int";
                                                                        } else if (return_type.equals("string")) {
                                                                                return_type = "String";
                                                                        }
                                                                        System.out.println("Method name: " +
method_name + " return type: " + return_type);

                                                                        Hashtable returnPList = (Hashtable)
m.getparameterList();
                                                                        int nParams = m.getnumberParameters();
                                                                        System.out.println("Number of params: " +
nParams);
                                                                        for(int j = 1;j <= nParams;j++) {
                                                                                Integer pid = new Integer(j);
                                                                                Parameter p = (Parameter)
returnPList.get(pid);
                                                                        System.out.println("*****************");
                                                                                String param_name =
p.getparameterName();
                                                                                String param_returntype =
p.getreturnType();
                                                                                if (param_returntype.equals("long")) {
                                                                                        param_returntype = "int";
                                                                                } else if
(param_returntype.equals("string")) {
                                                                                        param_returntype = "String";
                                                                                }
                                                                                System.out.println("Parameter name: "
+ param_name + " return type: " + param_returntype);
                                                                                if(j == nParams) {
                                                                                        parameters_signature += " "
+ param_returntype + " " + param_name;
                                                                                        parameters_name += " " +
param_name;
                                                                                }else {
                                                                                        parameters_signature += " "
+ param_returntype + " " + param_name + ",";
                                                                                        parameters_name += " " +
param_name + ",";
                                                                                }
                                                                        }
                                                                        System.out.println("parameters_signature: " +
parameters_signature + "parameters_name: " + parameters_name);
                                                                        if(return_type.equals("void")){
                                                                                parseVoidMethod(fileWriter);
                                                                        }
                                                                        else
                                                                                parseMethod(fileWriter);
                                                                }
                                                        }
                                                }//end of else
                                        } // end of WHILE loop
                                        fileWriter.close();
                                }catch(Exception e){
                                        System.out.println("Parse File Exception: " + e);
                                        e.printStackTrace();
                                }
                        }//end pf parseFile

                        public void parseMethod(BufferedWriter fileWriter) {
                                int mfilePosition;
                                String mtagContent;
```

```
                    String value_str = "null";
                    String value_int = "0";
                    String value_bool = "false";
                    String value_double = "0.0";

                    try {
                              BufferedReader mfileReader = new BufferedReader(new
FileReader("GlueCodeMethods_CS_RC_Template.txt"));
                              while( (mfilePosition=mfileReader.read()) >= 0 ) {
                                        //System.out.println(filePosition);
                                        if (mfilePosition != '<'){
                                                  fileWriter.write(mfilePosition);
                                        }else     {
                                                  mtagContent = "";
                                                  while( (mfilePosition=mfileReader.read()) != '>' ) {
                                                            mtagContent += (char)mfilePosition;
                                                  }
                                                  System.out.println(mtagContent);
                                                  // Compare the tagContents
                                                  if ( mtagContent.equals("method_name") ) {
                                                            fileWriter.write(method_name);

                                                  }else if ( mtagContent.equals("return_type") ) {
                                                            fileWriter.write(return_type);
                                                  }else if ( mtagContent.equals("parameters_signature") ) {
                                                            fileWriter.write(parameters_signature);
                                                  }else if ( mtagContent.equals("parameters_name") ) {
                                                            fileWriter.write(parameters_name);
                                                  }else if ( mtagContent.equals("value") ) {
                                                            if(return_type.equals("int")) {
                                                                      fileWriter.write(value_int);
                                                            }else if(return_type.equals("String"))  {
                                                                      fileWriter.write(value_str);
                                                            }else if(return_type.equals("boolean"))  {
                                                                      fileWriter.write(value_bool);
                                                            }else if(return_type.equals("double")) {
                                                                      fileWriter.write(value_double);
                                                            }
                                                  }
                                        }//end of else
                              } // end of WHILE loop
                              parameters_signature = "";
                              parameters_name = "";
                    }catch(Exception e){
                              System.out.println("Parse Method Exception: " + e);
                              e.printStackTrace();
                    }
          }//end pf parseMethod

public void parseVoidMethod(BufferedWriter fileWriter) {
                    int mfilePosition;
                    String mtagContent;

                    try {
                              BufferedReader mfileReader = new BufferedReader(new
FileReader("GlueCodeVoidMethods_CS_RC_Template.txt"));
                              while( (mfilePosition=mfileReader.read()) >= 0 ) {
                                        //System.out.println(filePosition);
                                        if (mfilePosition != '<'){
                                                  fileWriter.write(mfilePosition);
                                        }else     {
                                                  mtagContent = "";
                                                  while( (mfilePosition=mfileReader.read()) != '>' ) {
                                                            mtagContent += (char)mfilePosition;
                                                  }
                                                  System.out.println(mtagContent);
                                                  // Compare the tagContents
                                                  if ( mtagContent.equals("method_name") ) {
                                                            fileWriter.write(method_name);

                                                  }else if ( mtagContent.equals("return_type") ) {
                                                            fileWriter.write(return_type);
                                                  }else if ( mtagContent.equals("parameters_signature") ) {
                                                            fileWriter.write(parameters_signature);
                                                  }else if ( mtagContent.equals("parameters_name") ) {
                                                            fileWriter.write(parameters_name);
                                                  }
                                        }//end of else
                              } // end of WHILE loop
                              parameters_signature = "";
                              parameters_name = "";
                    }catch(Exception e){
                              System.out.println("Parse Method Exception: " + e);
                              e.printStackTrace();
                    }
          }//end pf parseVoidMethod
}//end of GlueCodeGen_CS_RC class
```

GlueCodeGenGUI_CS_RC.java

```
/** This class is used to compile and deploy the generated glue code
 *  required to interoperate between RMI and CORBA components
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.util.*;
import java.io.*;

public class GlueCodeGenGUI_CS_RC extends Thread implements Runnable{

        String GlueComponent;

        public GlueCodeGenGUI_CS_RC(String GlueComponent) {
                super();
                this.GlueComponent = GlueComponent;
        }

        public void run() {
                try {
                        Runtime rt = Runtime.getRuntime();

                        // To compile GlueComponents
                        Process proc3 = rt.exec("cmd /c start javac -classpath
%ORB2%/lib/orb2.jar;C:/Kalpana/xmlParser/xerces.jar;C:/Kalpana/GG;C:/Kalpana/SuperBank1/CashVS/CPP " +
GlueComponent + ".java");
                        InputStreamReader isr3 = new InputStreamReader(proc3.getInputStream());
                        BufferedReader br3 = new BufferedReader(isr3);
                        String line3 = null;
                        while ( (line3 = br3.readLine()) != null)
                                System.out.println(line3);
                        int exitVal3 = proc3.waitFor();
                        if(exitVal3==0)
                                System.out.println("GlueCodeGenGUI_CS_RC All Files: Successfully
Compiled. ");
                        else
                                System.out.println("GlueCodeGenGUI_CS_RC Compiled with errors. ");

                        // To rmic GlueComponent
                        Process proc4 = rt.exec("cmd /c start rmic " + GlueComponent);
                        InputStreamReader isr4 = new InputStreamReader(proc4.getInputStream());
                        BufferedReader br4 = new BufferedReader(isr4);
                        String line4 = null;
                        while ( (line4 = br4.readLine()) != null)
                                System.out.println(line4);
                        int exitVal4 = proc4.waitFor();
                        if(exitVal4==0)
                                System.out.println("GlueCodeGenGUI_CS_RC Successfully Compiled. ");
                        else
                                System.out.println("GlueCodeGenGUI_CS_RC Compiled with errors. ");

                        // To execute GlueComponent
                        Process proc5 = rt.exec("cmd /c start java -Djava.security.policy=policy -
Djava.rmi.server.codebase=file:/C:/Kalpana/GG/ -classpath
%ORB2%/lib/orb2.jar;C:/Kalpana/GG;C:/Kalpana/SuperBank1/CashVS/CPP " + GlueComponent + " -ORBInitRef
NameService=corbaloc::134.68.140.101:9999/NameService");
                        InputStreamReader isr5 = new InputStreamReader(proc5.getInputStream());
                        BufferedReader br5 = new BufferedReader(isr5);
                        String line5 = null;
                        while ( (line5 = br5.readLine()) != null)
                                System.out.println(line5);
                        int exitVal5 = proc5.waitFor();
                        if(exitVal5==0)
                                System.out.println("GlueCodeGenGUI_CS_RC File: " + GlueComponent + "
Successfully executed. ");
                        else
                                System.out.println("GlueCodeGenGUI_CS_RC File: " + GlueComponent + "
Executed with errors. ");
                } catch (Throwable t) {
                        t.printStackTrace();
                }
        }
}
```

GlueCodeGen_RS_CC.java

```
/** This class is used to generate the glue code required to
 *  interoperate between CORBA and RMI components using the templates
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class GlueCodeGen_RS_CC extends Thread implements Runnable{

//      Hashtable compList;
        Component comp;
        String RSComponent_name;
        String RSInterface_name;
```

```
		String RSComponent_location;
		String method_name;
		String return_type;
		String parameters_signature;
		String parameters_name;
		String GLUECompName;
		String init_resp_pair;

		public GlueCodeGen_RS_CC(Component comp, String init_resp_pair) {
			super();
			this.comp = comp;
			this.init_resp_pair = init_resp_pair;
		}

		public void run() {

			//GETTING & SETTING VALUES

			RSComponent_name = comp.getcomponentName();
			System.out.println("RSComponent_name: " + RSComponent_name);

			RSInterface_name = comp.getprovidedinterfaceName();
			System.out.println("RSInterface_name: " + RSInterface_name);
			GLUECompName = "Glue_RS_CC_" + RSInterface_name.substring(0,4);

			RSComponent_location = comp.getcomponentLocation();
			System.out.println("RSComponent_location: " + RSComponent_location);

			System.out.println("**********************************************************");
			method_name = "";
			return_type = "";
			parameters_signature = "";
			parameters_name = "";
			parseFile();
			GlueCodeGenGUI_RS_CC GCGGUI = new GlueCodeGenGUI_RS_CC(GLUECompName,
RSInterface_name.substring(0,4));
			GCGGUI.start();
		}

		// parses the template and writes into Glue code file
		public void parseFile() {
			int filePosition;
			String tagContent;
			try {
				BufferedReader fileReader = new BufferedReader(new
FileReader("GlueCode_RS_CC_Template.txt"));
				BufferedWriter fileWriter = new BufferedWriter(new FileWriter(GLUECompName +
".java"));

				Hashtable returnMList = (Hashtable) comp.getmethodList();
				int nMethods = comp.getnumberMethods();
				System.out.println("******************************");
				System.out.println("Number of methods: " + nMethods);

				while( (filePosition=fileReader.read()) >= 0 ) {
					//System.out.println(filePosition);
					if (filePosition != '<'){
						fileWriter.write(filePosition);
					}else	{
						tagContent = "";
						while( (filePosition=fileReader.read()) != '>' ) {
							tagContent += (char)filePosition;
```

```
                                        }
                                        System.out.println(tagContent);
                                        // Compare the tagContents
                                        if ( tagContent.equals("Glue_RS_CC") ) {
                                                fileWriter.write(GLUECompName);
                                        }else if ( tagContent.equals("RSInterface_name") ) {
                                                fileWriter.write(RSInterface_name);
                                        }else if ( tagContent.equals("init_resp_pair") ) {
                                                fileWriter.write(init_resp_pair);
                                        }else if ( tagContent.equals("METHODS") ) {
                                                for(int i=1;i <= nMethods;i++) {
                                                        Integer mid = new Integer(i);
                                                        Method m = (Method) returnMList.get(mid);
                                        System.out.println("******************************");
                                                        method_name = m.getmethodName();
                                                        return_type = m.getreturnType();
                                                        System.out.println("Method name: " +
method_name + " return type: " + return_type);

                                                        Hashtable returnPList = (Hashtable)
m.getparameterList();
                                                        int nParams = m.getnumberParameters();
                                                        System.out.println("Number of params: " +
nParams);
                                                        for(int j = 1;j <= nParams;j++) {
                                                                Integer pid = new Integer(j);
                                                                Parameter p = (Parameter)
returnPList.get(pid);
                                                        System.out.println("******************");
                                                                String param_name =
p.getparameterName();
                                                                String param_returntype =
p.getreturnType();
                                                                System.out.println("Parameter name: "
+ param_name + " return type: " + param_returntype);
                                                                if(j == nParams) {
                                                                        parameters_signature += " "
+ param_returntype + " " + param_name;
                                                                        parameters_name += " " +
param_name;
                                                                }else {
                                                                        parameters_signature += " "
+ param_returntype + " " + param_name + ",";
                                                                        parameters_name += " " +
param_name + ",";
                                                                }

                                                        }
                                                        System.out.println("parameters_signature: " +
parameters_signature + "parameters_name: " + parameters_name);
                                                        if(return_type.equals("void")){
                                                                parseVoidMethod(fileWriter);
                                                        }
                                                        else
                                                                parseMethod(fileWriter);
                                                }
                                        }
                                }//end of else
                        } // end of WHILE loop
                        fileWriter.close();
                }catch(Exception e){
                        System.out.println("Parse File Exception: " + e);
                        e.printStackTrace();
                }
        }//end pf parseFile

        public void parseMethod(BufferedWriter fileWriter) {
                int mfilePosition;
                String mtagContent;
                String value_str = "null";
                String value_int = "0";
                String value_bool = "false";
                String value_double = "0.0";
                try {
                        BufferedReader mfileReader = new BufferedReader(new
FileReader("GlueCodeMethods_RS_CC_Template.txt"));
                        while( (mfilePosition=mfileReader.read()) >= 0 ) {
                                //System.out.println(filePosition);
                                if (mfilePosition != '<'){
                                        fileWriter.write(mfilePosition);
                                }else   {
                                        mtagContent = "";
                                        while( (mfilePosition=mfileReader.read()) != '>' ) {
                                                mtagContent += (char)mfilePosition;
                                        }
                                        System.out.println(mtagContent);
                                        // Compare the tagContents
                                        if ( mtagContent.equals("method_name") ) {
                                                fileWriter.write(method_name);

                                        }else if ( mtagContent.equals("return_type") ) {
                                                fileWriter.write(return_type);
                                        }else if ( mtagContent.equals("parameters_signature") ) {
                                                fileWriter.write(parameters_signature);
                                        }else if ( mtagContent.equals("parameters_name") ) {
                                                fileWriter.write(parameters_name);
                                        }else if ( mtagContent.equals("value") ) {
                                                if(return_type.equals("int")) {
                                                        fileWriter.write(value_int);
                                                }else if(return_type.equals("String"))  {
                                                        fileWriter.write(value_str);
                                                }else if(return_type.equals("boolean"))  {
                                                        fileWriter.write(value_bool);
                                                }else if(return_type.equals("double")) {
                                                        fileWriter.write(value_double);
                                                }
                                        }
                                }//end of else
                        } // end of WHILE loop
                        parameters_signature = "";
                        parameters_name = "";
                }catch(Exception e){
                        System.out.println("Parse Method Exception: " + e);
                        e.printStackTrace();
                }
        }//end pf parseMethod

        public void parseVoidMethod(BufferedWriter fileWriter) {
                int mfilePosition;
                String mtagContent;
                try {
                        BufferedReader mfileReader = new BufferedReader(new
FileReader("GlueCodeVoidMethods_RS_CC_Template.txt"));
                        while( (mfilePosition=mfileReader.read()) >= 0 ) {
```

```
                                        //System.out.println(filePosition);
                                        if (mfilePosition != '<'){
                                                fileWriter.write(mfilePosition);
                                        }else   {
                                                mtagContent = "";
                                                while( (mfilePosition=mfileReader.read()) != '>' ) {
                                                        mtagContent += (char)mfilePosition;
                                                }
                                                System.out.println(mtagContent);
                                                // Compare the tagContents
                                                if ( mtagContent.equals("method_name") ) {
                                                        fileWriter.write(method_name);

                                                }else if ( mtagContent.equals("return_type") ) {
                                                        fileWriter.write(return_type);
                                                }else if ( mtagContent.equals("parameters_signature") ) {
                                                        fileWriter.write(parameters_signature);
                                                }else if ( mtagContent.equals("parameters_name") ) {
                                                        fileWriter.write(parameters_name);
                                                }
                                        }//end of else
                                } // end of WHILE loop
                                parameters_signature = "";
                                parameters_name = "";
                        }catch(Exception e){
                                System.out.println("Parse Method Exception: " + e);
                                e.printStackTrace();
                        }
                }//end pf parseMethod
}//end of GlueCodeGen_RS_CC class
```

GlueCodeGenGUI_RS_CC.java

```
/** This class is used to compile and deploy the generated glue code
 *  required to interoperate between CORBA and RMI components
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.util.*;
import java.io.*;

public class GlueCodeGenGUI_RS_CC extends Thread implements Runnable{

        String GlueComponent;
        String GlueInterface;

        public GlueCodeGenGUI_RS_CC(String GlueComponentName, String GlueInterfaceName) {
                GlueComponent = GlueComponentName;
                GlueInterface = GlueInterfaceName;
        }

        public void run() {
                try {
                        Runtime rt = Runtime.getRuntime();

                        // To compile GlueComponents
                        Process proc3 = rt.exec("cmd /c start javac -classpath
%ORB2%/lib/orb2.jar;C:/Kalpana/xmlParser/xerces.jar;C:/Kalpana/GG;C:/Kalpana/SuperBank1 " + GlueComponent
+ ".java");
```

```
                                InputStreamReader isr3 = new InputStreamReader(proc3.getInputStream());
                                BufferedReader br3 = new BufferedReader(isr3);
                                String line3 = null;
                                while ( (line3 = br3.readLine()) != null)
                                        System.out.println(line3);
                                int exitVal3 = proc3.waitFor();
                                System.out.println("GlueCodeGenGUI_RS_CC " + GlueComponent + " exitVal3 " +
exitVal3);
                                if(exitVal3==0)
                                        System.out.println("GlueCodeGenGUI_RS_CC All Files: Successfully
Compiled. ");
                                else
                                        System.out.println("GlueCodeGenGUI_RS_CC Compiled with errors. ");

                        int y = System.in.read();

                                // To execute GlueComponent
                                Process proc5 = rt.exec("cmd /c start java -Djava.security.policy=policy -
Djava.rmi.server.codebase=file:/C:/Kalpana/GG/ -classpath C:/Kalpana/GG;C:/Kalpana/SuperBank1 " +
GlueComponent + " -ORBInitRef NameService=corbaloc::134.68.140.101:9999/NameService");
                                InputStreamReader isr5 = new InputStreamReader(proc5.getInputStream());
                                BufferedReader br5 = new BufferedReader(isr5);
                                String line5 = null;
                                while ( (line5 = br5.readLine()) != null)
                                        System.out.println(line5);
                                int exitVal5 = proc5.waitFor();
                                if(exitVal5==0)
                                        System.out.println("GlueCodeGenGUI_RS_CC File: " + GlueComponent + "
Successfully executed. ");
                                else
                                        System.out.println("GlueCodeGenGUI_RS_CC File: " + GlueComponent + "
Executed with errors. ");
                        } catch (Throwable t) {
                                t.printStackTrace();
                        }
        }
}
```

GlueConfigureGen_CS_RC.java

```
/** This class is used to generate the glue configure code required to
 *  configure RMI to Glue component and Glue to CORBA component using templates
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class GlueConfigureGen_CS_RC extends Thread implements Runnable{

        Component CScomp;
        Component RCcomp;
        Component GLUEcomp;
        String CSid;
        String CSkind;
        String CSComponentName;
        String RCLocation;
        String GLUELocation;
	String RCInterface;
	String GLUEInterface;
	String GLUEConfCompName;
	String init_resp_pair;

	public GlueConfigureGen_CS_RC(Component comp1, Component comp2, String init_resp_pair) {
		super();
		this.RCcomp = comp1;
		this.CScomp = comp2;
		this.init_resp_pair = init_resp_pair;
	}

	public void run() {
		GLUEcomp = new Component();

		GLUEcomp.setprovidedinterfaceName(CScomp.getprovidedinterfaceName());
		GLUEcomp.setcomponentLocation("//134.68.140.101:9000/Glue_CS_RC_" +
CScomp.getprovidedinterfaceName().substring(0,4));

		//GETTING VALUES

		System.out.println("*********************************************************");
		CSid = CScomp.getcomponentid();
		System.out.println("CSid: " + CSid);
		CSkind = CScomp.getcomponentkind();
		System.out.println("CSkind: " + CSkind);
		RCInterface = RCcomp.getprovidedinterfaceName();
		System.out.println("RCInterface: " + RCInterface);
		RCLocation = RCcomp.getcomponentLocation();
		System.out.println("RCLocation: " + RCLocation);
		GLUEInterface = GLUEcomp.getprovidedinterfaceName();
		System.out.println("GLUEInterface: " + GLUEInterface);
		GLUEConfCompName = "Configure_CS_RC_" + GLUEInterface.substring(0,4);
		GLUELocation = GLUEcomp.getcomponentLocation();
		System.out.println("GLUELocation: " + GLUELocation);
		CSComponentName = CScomp.getcomponentName();

		System.out.println("parseFile being called");
		parseFile();
		GlueConfigureGenGUI_CS_RC GCfGGUI = new
GlueConfigureGenGUI_CS_RC(GLUEConfCompName);
		GCfGGUI.start();
	}

	public void parseFile() {
		System.out.println("Inside parseFile");
		int filePosition;
		String tagContent;
		try {
			BufferedReader fileReader = new BufferedReader(new
FileReader("GlueConfigure_CS_RC_Template.txt"));
			BufferedWriter fileWriter = new BufferedWriter(new FileWriter(GLUEConfCompName
+ ".java"));

			while( (filePosition=fileReader.read()) >= 0 ) {
				//System.out.println(filePosition);
				if (filePosition != '<'){
					fileWriter.write(filePosition);
				}else	{
					tagContent = "";
					while( (filePosition=fileReader.read()) != '>' ) {
						tagContent += (char)filePosition;
```

```
                                                }
                                                System.out.println(tagContent);
                                                // Compare the tagContents
                                                if ( tagContent.equals("Configure_CS_RC") ) {
                                                        fileWriter.write(GLUEConfCompName);
                                                }else if ( tagContent.equals("CSid") ) {
                                                        fileWriter.write(CSid);
                                                }else if ( tagContent.equals("CSkind") ) {
                                                        fileWriter.write(CSkind);
                                                }else if ( tagContent.equals("RCLocation") ) {
                                                        fileWriter.write(RCLocation);
                                                }else if ( tagContent.equals("GLUELocation") ) {
                                                        fileWriter.write(GLUELocation);
                                                }else if ( tagContent.equals("RCInterface") ) {
                                                        fileWriter.write(RCInterface);
                                                }else if ( tagContent.equals("GLUEInterface") ) {
                                                        fileWriter.write(GLUEInterface);
                                                }else if ( tagContent.equals("CSComponentName") ) {
                                                        fileWriter.write(CSComponentName);
                                                }else if ( tagContent.equals("init_resp_pair") ) {
                                                        fileWriter.write(init_resp_pair);
                                                }
                                        }//end of else
                                } // end of WHILE loop
                                fileWriter.close();
                        }catch(Exception e){
                                System.out.println("Parse File Exception: " + e);
                                e.printStackTrace();
                        }
                }//end pf parseFile
}//end of GlueConfigureGen_CS_RC class
```

GlueConfigureGenGUI_CS_RC.java

```
/** This class is used to compile and deploy the generated glue configure code
 *  required to interoperate between RMI and CORBA components
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.util.*;
import java.io.*;

public class GlueConfigureGenGUI_CS_RC extends Thread implements Runnable{
        String GlueConfigureComponent;

        public GlueConfigureGenGUI_CS_RC(String GlueConfigureComponent){
                super();
                this.GlueConfigureComponent = GlueConfigureComponent;
        }

        public void run() {
                try {
                        Runtime rt = Runtime.getRuntime();

                        // To compile GlueConfigureComponents
                        Process proc3 = rt.exec("cmd /c start javac -classpath
%ORB2%/lib/orb2.jar;C:/Kalpana/xmlParser/xerces.jar;C:/Kalpana/GG;C:/Kalpana/SuperBank1 " +
GlueConfigureComponent + ".java");
                        InputStreamReader isr3 = new InputStreamReader(proc3.getInputStream());
```

```
                                BufferedReader br3 = new BufferedReader(isr3);
                                String line3 = null;
                                while ( (line3 = br3.readLine()) != null)
                                        System.out.println(line3);
                                int exitVal3 = proc3.waitFor();
                                if(exitVal3==0)
                                        System.out.println("GlueConfigureGenGUI_CS_RC File: " +
GlueConfigureComponent + ".java Successfully Compiled. ");
                                else
                                        System.out.println("GlueConfigureGenGUI_CS_RC File: " +
GlueConfigureComponent + ".java Compiled with errors. ");

                                // To execute GlueConfigureComponent
                                Process proc4 = rt.exec("cmd /c start java -Djava.security.policy=policy -classpath
C:/Kalpana/GG;C:/Kalpana/SuperBank1 " + GlueConfigureComponent);
                                InputStreamReader isr4 = new InputStreamReader(proc4.getInputStream());
                                BufferedReader br4 = new BufferedReader(isr4);
                                String line4 = null;
                                while ( (line4 = br4.readLine()) != null)
                                        System.out.println(line4);
                                int exitVal4 = proc4.waitFor();
                                if(exitVal4==0)
                                        System.out.println("GlueConfigureGenGUI_CS_RC File: " +
GlueConfigureComponent + ".java Successfully executed. ");
                                else
                                        System.out.println("GlueConfigureGenGUI_CS_RC File: " +
GlueConfigureComponent + ".java executed with errors. ");
                        } catch (Throwable t) {
                                t.printStackTrace();
                        }
        }
}
```

GlueConfigureGen_RS_CC.java

```
/** This class is used to generate the glue configure code required to
 *  configure CORBA to Glue component and Glue to RMI component using templates
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class GlueConfigureGen_RS_CC extends Thread implements Runnable{

        Component CCcomp;
        Component RScomp;
        Component GLUEcomp = new Component();
        String CCid;
        String CCkind;
        String RSLocation;
        String RSComponentName;
        String GLUEid;
        String GLUEkind;
        String CCInterface_name;
        String GLUEInterface_name;
        String GLUEConfCompName;
        String init_resp_pair;
	public GlueConfigureGen_RS_CC(Component comp1, Component comp2, String init_resp_pair) {
		super();
		this.CCcomp = comp1;
		this.RScomp = comp2;
		this.init_resp_pair = init_resp_pair;
	}

	public void run() {

		//SETTING VALUES
		GLUEcomp.setcomponentid("Glue_RS_CC_" + 
RScomp.getprovidedinterfaceName().substring(0,4));
		GLUEcomp.setcomponentkind("");
		GLUEcomp.setprovidedinterfaceName(RScomp.getprovidedinterfaceName());

		//GETTING VALUES
	System.out.println("*********************************************************");
		CCid = CCcomp.getcomponentid();
		System.out.println("CCid: " + CCid);
		CCkind = CCcomp.getcomponentkind();
		System.out.println("CCkind: " + CCkind);
		GLUEid = GLUEcomp.getcomponentid();
		System.out.println("GLUEid: " + GLUEid);
		GLUEkind = GLUEcomp.getcomponentkind();
		System.out.println("GLUEkind: " + GLUEkind);
		RSLocation = RScomp.getcomponentLocation();
		System.out.println("RSLocation: " + RSLocation);
		RSComponentName = RScomp.getcomponentName();
		System.out.println("RSComponentName: " + RSComponentName);
		CCInterface_name = CCcomp.getprovidedinterfaceName();
		System.out.println("CCInterface_name: " + CCInterface_name);
		GLUEInterface_name = GLUEcomp.getprovidedinterfaceName();
		System.out.println("GLUEInterface_name: " + GLUEInterface_name);
		GLUEConfCompName = "Configure_RS_CC_" + GLUEInterface_name.substring(0,4);

		System.out.println("parseFile being called");
		parseFile();
		GlueConfigureGenGUI_RS_CC GCfGGUI = new 
GlueConfigureGenGUI_RS_CC(GLUEConfCompName);
		GCfGGUI.start();
	}

	public void parseFile() {
		System.out.println("Inside parseFile");
		int filePosition;
		String tagContent;
		try {
			BufferedReader fileReader = new BufferedReader(new 
FileReader("GlueConfigure_RS_CC_Template.txt"));
			BufferedWriter fileWriter = new BufferedWriter(new FileWriter(GLUEConfCompName 
+ ".java"));

			while( (filePosition=fileReader.read()) >= 0 ) {
				//System.out.println(filePosition);
				if (filePosition != '<'){
					fileWriter.write(filePosition);
				}else	{
					tagContent = "";
					while( (filePosition=fileReader.read()) != '>' ) {
						tagContent += (char)filePosition;
					}
```

```
                                                System.out.println(tagContent);
                                                // Compare the tagContents
                                                if ( tagContent.equals("Configure_RS_CC") ) {
                                                        fileWriter.write(GLUEConfCompName);
                                                }else if ( tagContent.equals("CCid") ) {
                                                        fileWriter.write(CCid);
                                                }else if ( tagContent.equals("CCkind") ) {
                                                        fileWriter.write(CCkind);
                                                }else if ( tagContent.equals("RSLocation") ) {
                                                        fileWriter.write(RSLocation);
                                                }else if ( tagContent.equals("RSComponentName") ) {
                                                        fileWriter.write(RSComponentName);
                                                }else if ( tagContent.equals("GLUEid") ) {
                                                        fileWriter.write(GLUEid);
                                                }else if ( tagContent.equals("GLUEkind") ) {
                                                        fileWriter.write(GLUEkind);
                                                }else if ( tagContent.equals("CCInterface_name") ) {
                                                        fileWriter.write(CCInterface_name);
                                                }else if ( tagContent.equals("GLUEInterface_name") ) {
                                                        fileWriter.write(GLUEInterface_name);
                                                }else if ( tagContent.equals("init_resp_pair") ) {
                                                        fileWriter.write(init_resp_pair);
                                                }
                                        }//end of else
                                } // end of WHILE loop
                                fileWriter.close();
                        }catch(Exception e){
                                System.out.println("Parse File Exception: " + e);
                                e.printStackTrace();
                        }
                }//end pf parseFile

}//end of GlueConfigureGen_RS_CC class
```

GlueConfigureGenGUI_RS_CC.java

```
/** This class is used to compile and deploy the generated glue configure code
 *  required to interoperate between CORBA and RMI components
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.util.*;
import java.io.*;

public class GlueConfigureGenGUI_RS_CC extends Thread implements Runnable{
        String GlueConfigureComponent;

        public GlueConfigureGenGUI_RS_CC(String GlueConfigureComponent){
                super();
                this.GlueConfigureComponent = GlueConfigureComponent;
        }
        public void run() {
                try {
                        Runtime rt = Runtime.getRuntime();

                        // To compile GlueConfigureComponents
                        Process proc3 = rt.exec("cmd /c start javac -classpath
%ORB2%/lib/orb2.jar;C:/Kalpana/xmlParser/xerces.jar;C:/Kalpana/GG;C:/Kalpana/SuperBank1/ATM " +
GlueConfigureComponent + ".java");
```

```
				InputStreamReader isr3 = new InputStreamReader(proc3.getInputStream());
				BufferedReader br3 = new BufferedReader(isr3);
				String line3 = null;
				while ( (line3 = br3.readLine()) != null)
					System.out.println(line3);
				int exitVal3 = proc3.waitFor();
				if(exitVal3==0)
					System.out.println("GlueConfigureGenGUI_RS_CC File: " +
GlueConfigureComponent + ".java Successfully Compiled. ");
				else
					System.out.println("GlueConfigureGenGUI_RS_CC File: " +
GlueConfigureComponent + ".java Compiled with errors. ");

				// To execute GlueConfigureComponent
				Process proc4 = rt.exec("cmd /c start java -Djava.security.policy=policy -
Djava.rmi.server.codebase=file:/C:/Kalpana/GG/ -classpath C:/Kalpana/GG;C:/Kalpana/SuperBank1/ATM " +
GlueConfigureComponent + " -ORBInitRef NameService=corbaloc::134.68.140.101:9999/NameService");
				InputStreamReader isr4 = new InputStreamReader(proc4.getInputStream());
				BufferedReader br4 = new BufferedReader(isr4);
				String line4 = null;
				while ( (line4 = br4.readLine()) != null)
					System.out.println(line4);
				int exitVal4 = proc4.waitFor();
				if(exitVal4==0)
					System.out.println("GlueConfigureGenGUI_RS_CC File: " +
GlueConfigureComponent + ".java Successfully executed. ");
				else
					System.out.println("GlueConfigureGenGUI_RS_CC File: " +
GlueConfigureComponent + ".java executed with errors. ");
			} catch (Throwable t) {
				t.printStackTrace();
			}
		}
}
```

UMMSpecificationParser.java

```
/** This class is used to parse the UMM specification XML document
 *  to obtain the details of the component and store them in component classes
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.IOException;
import org.w3c.dom.Document;
import org.w3c.dom.DocumentType;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.apache.xerces.parsers.DOMParser;
import java.util.*;

public class UMMSpecificationParser {

	private Component component;

	public UMMSpecificationParser(String url) {
		DOMParser parser = new DOMParser();
		try {
			parser.parse(url);
```

```
                        Document doc = parser.getDocument();
                        NodeList children = doc.getChildNodes();
                        if (children != null) {
                                for (int i = 0; i < children.getLength(); i++) {
                                        if(children.item(i).getNodeName().equalsIgnoreCase("Component")) {
                                                parseUMMSpecification(children.item(i));
                                                break;
                                        }
                                }
                        }
                } catch (IOException e) {
                        System.out.println("Error reading URL: " + e.getMessage());
                } catch (Exception ex) {
                        System.out.println("Error in parsing: " + ex.getMessage());
                }
        }

        public Component getComponent() {
                return component;
        }

        private void parseUMMSpecification(Node node) {
                NodeList children = node.getChildNodes();
                if(children == null)
                        return;
                component = new Component();

                for (int i = 0; i < children.getLength(); i++) {
                        if(children.item(i).getNodeName().equalsIgnoreCase("componentName")) {
                        component.setcomponentName(children.item(i).getFirstChild().getNodeValue().trim());
                        } else if(children.item(i).getNodeName().equalsIgnoreCase("componentLocation")) {
                       component.setcomponentLocation(children.item(i).getFirstChild().getNodeValue().trim());
                        } else if(children.item(i).getNodeName().equalsIgnoreCase("componentid")) {
                        component.setcomponentid(children.item(i).getFirstChild().getNodeValue().trim());
                        } else if(children.item(i).getNodeName().equalsIgnoreCase("componentkind")) {
                        component.setcomponentkind(children.item(i).getFirstChild().getNodeValue().trim());
                        } else if(children.item(i).getNodeName().equalsIgnoreCase("componentTechnology")) {
                component.setcomponentTechnology(children.item(i).getFirstChild().getNodeValue().trim());
                        } else if(children.item(i).getNodeName().equalsIgnoreCase("providedinterface")) {
                                NodeList children_providedinterface = children.item(i).getChildNodes();
                                if(children_providedinterface == null)
                                        break;
                                for(int j = 0; j < children_providedinterface.getLength(); j++) {
                        if(children_providedinterface.item(j).getNodeName().equals("interfaceName")) {
        component.setprovidedinterfaceName(children_providedinterface.item(j).getFirstChild().getNodeValue().trim());
                                        } else
if(children_providedinterface.item(j).getNodeName().equals("packageName")) {
        component.setpackageName(children_providedinterface.item(j).getFirstChild().getNodeValue().trim());
                                        } else
if(children_providedinterface.item(j).getNodeName().equals("methodList")) {
                                                int mid = 1;
                                                NodeList children_methodList =
children_providedinterface.item(j).getChildNodes();
                                                Hashtable mList = new Hashtable();
                                                for(int k = 0; k < children_methodList.getLength(); k++) {
                        if(children_methodList.item(k).getNodeName().equalsIgnoreCase("numberMethods")) {
        component.setnumberMethods(Integer.parseInt(children_methodList.item(k).getFirstChild().getNodeValue().trim()));
                                                        } else
if(children_methodList.item(k).getNodeName().equalsIgnoreCase("Method")) {
                                                                Method method = new Method();
                                                                NodeList children_Method =
children_methodList.item(k).getChildNodes();
```

```
                                                                                    for(int l = 0; l < children_Method.getLength();
l++) {
                                        if(children_Method.item(l).getNodeName().equalsIgnoreCase("methodName")) {
                                method.setmethodName(children_Method.item(l).getFirstChild().getNodeValue().trim());
                                                                                            } else
if(children_Method.item(l).getNodeName().equalsIgnoreCase("returnType")) {
                                        method.setreturnType(children_Method.item(l).getFirstChild().getNodeValue().trim());
                                                                                            } else
if(children_Method.item(l).getNodeName().equalsIgnoreCase("parameterList")) {
                                                                                                    int pid = 1;
                                                                                                    NodeList
children_parameterList = children_Method.item(l).getChildNodes();
                                                                                                    Hashtable pList = new
Hashtable();
                                                                                                    for(int m = 0; m <
children_parameterList.getLength(); m++) {
                                if(children_parameterList.item(m).getNodeName().equalsIgnoreCase("numberParameters")) {
                        method.setnumberParameters(Integer.parseInt(children_parameterList.item(m).getFirstChild().getNodeValue().trim()));
                                                                                                            } else
if(children_parameterList.item(m).getNodeName().equalsIgnoreCase("Parameter")) {
                        Parameter parameter = new Parameter();
                        NodeList children_Parameter = children_parameterList.item(m).getChildNodes();
                        for(int n = 0; n < children_Parameter.getLength(); n++) {
                                if(children_Parameter.item(n).getNodeName().equalsIgnoreCase("parameterName")) {
                                  parameter.setparameterName(children_Parameter.item(n).getFirstChild().getNodeValue().trim());
                                } else if(children_Parameter.item(n).getNodeName().equalsIgnoreCase("returnType")) {
                                parameter.setreturnType(children_Parameter.item(n).getFirstChild().getNodeValue().trim());
                                }
                        }//end of for
                        pList.put(new Integer(pid), parameter);
                        pid++;
}//end of else if
}//end of for
                    method.setparameterList(pList);
}//end of else if
}//end of for
//method.setparameterList(pList);
                 mList.put(new Integer(mid), method);
                 mid++;
}//end of else if
}//end of for
                component.setmethodList(mList);
}//end of else if
}//end of for
}//end of else if
                else if(children.item(i).getNodeName().equalsIgnoreCase("requiredinterface")) {
                NodeList children_requiredinterface = children.item(i).getChildNodes();
                if(children_requiredinterface == null)
                      break;
                      for(int j = 0; j < children_requiredinterface.getLength(); j++) {
                                if(children_requiredinterface.item(j).getNodeName().equals("interfaceName")) {
                      component.setrequiredinterfaceName(children_requiredinterface.item(j).getFirstChild().getNodeValue().trim());
                                } else if(children_requiredinterface.item(j).getNodeName().equals("packageName")) {
                      component.setpackageName(children_requiredinterface.item(j).getFirstChild().getNodeValue().trim());
                      } else if(children_requiredinterface.item(j).getNodeName().equals("methodList")) {
                      int mid = 1;
                      NodeList children_methodList = children_requiredinterface.item(j).getChildNodes();
                      Hashtable mList = new Hashtable();
                      for(int k = 0; k < children_methodList.getLength(); k++) {
                                if(children_methodList.item(k).getNodeName().equalsIgnoreCase("numberMethods")) {
                      component.setnumberMethods(Integer.parseInt(children_methodList.item(k).getFirstChild().getNodeValue().trim()));
                                        } else if(children_methodList.item(k).getNodeName().equalsIgnoreCase("Method")) {
```

```
                                                                                Method method = new Method();
                                                                                NodeList children_Method =
children_methodList.item(k).getChildNodes();
        for(int l = 0; l < children_Method.getLength(); l++) {
                if(children_Method.item(l).getNodeName().equalsIgnoreCase("methodName")) {
                method.setmethodName(children_Method.item(l).getFirstChild().getNodeValue().trim());
                } else if(children_Method.item(l).getNodeName().equalsIgnoreCase("returnType")) {
                method.setreturnType(children_Method.item(l).getFirstChild().getNodeValue().trim());
                } else if(children_Method.item(l).getNodeName().equalsIgnoreCase("parameterList")) {
                int pid = 1;
                NodeList children_parameterList = children_Method.item(l).getChildNodes();
                Hashtable pList = new Hashtable();
                for(int m = 0; m < children_parameterList.getLength(); m++) {
        if(children_parameterList.item(m).getNodeName().equalsIgnoreCase("numberParameters")) {
        method.setnumberParameters(Integer.parseInt(children_parameterList.item(m).getFirstChild().getNodeValue().trim()));
if(children_parameterList.item(m).getNodeName().equalsIgnoreCase("Parameter")) {
        Parameter parameter = new Parameter();
        NodeList children_Parameter = children_parameterList.item(m).getChildNodes();
        for(int n = 0; n < children_Parameter.getLength(); n++) {
        if(children_Parameter.item(n).getNodeName().equalsIgnoreCase("parameterName")) {
                parameter.setparameterName(children_Parameter.item(n).getFirstChild().getNodeValue().trim());
        } else if(children_Parameter.item(n).getNodeName().equalsIgnoreCase("returnType")) {
                parameter.setreturnType(children_Parameter.item(n).getFirstChild().getNodeValue().trim());
        }
        }//end of for
        pList.put(new Integer(pid), parameter);
        pid++;
                                                                                                    }//end of else if

                                                                                            }//end of for
        method.setparameterList(pList);
                                                                                        }//end of else if
                                                                                }//end of for
                                                                                //method.setparameterList(pList);
                                                                                mList.put(new Integer(mid), method);
                                                                                mid++;
                                                                        }//end of else if
                                                                }//end of for
                                                                component.setmethodList(mList);
                                                        }//end of else if
                                                }//end of for
                                        }//end of else if
                                }//end of for
                        }//end of method
}//end of class
```

Entity Classes

Component.java

```
/** This class is used to store the details of the components
 *  after parsing the UMM specification XML document
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class Component {

        String componentName;
        String providedinterfaceName;
        String requiredinterfaceName;
        String packageName;
        String componentLocation;
        String componentid;
        String componentkind;
        String componentTechnology;

        Hashtable methodList;
        int numberMethods;

        public Component() {
                super();
        }
        public void setcomponentName(String cName) {
                componentName = cName;
        }
        public String getcomponentName() {
                return componentName;
        }
        public void setprovidedinterfaceName(String piName) {
                providedinterfaceName = piName;
        }
        public String getprovidedinterfaceName() {
                return providedinterfaceName;
        }
        public void setrequiredinterfaceName(String riName) {
                requiredinterfaceName = riName;
        }
        public String getrequiredinterfaceName() {
                return requiredinterfaceName;
        }
        public void setpackageName(String pName) {
                packageName = pName;
        }
        public String getpackageName() {
                return packageName;
        }
        public void setcomponentLocation(String cLoc) {
                componentLocation = cLoc;
        }
        public String getcomponentLocation() {
                return componentLocation;
        }
        public void setcomponentid(String id) {
                componentid = id;
        }
        public String getcomponentid() {
                return componentid;
        }
        public void setcomponentkind(String kind) {
                componentkind = kind;
        }
        public String getcomponentkind() {
                return componentkind;
        }
```

```
		public void setcomponentTechnology(String tech) {
			componentTechnology = tech;
		}
		public String getcomponentTechnology() {
			return componentTechnology;
		}
		public void setmethodList(Hashtable mList) {
			methodList = mList;
		}
		public Hashtable getmethodList() {
			return methodList;
		}
		public void setnumberMethods(int n) {
			numberMethods = n;
		}
		public int getnumberMethods() {
			return numberMethods;
		}
}
```

Method.java

```
/** This class is used to store the details of the methods of components
 *  after parsing the UMM specification XML document
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class Method {

		String methodName;
		String returnType;

		Hashtable parameterList;
		int numberParameters;

		public Method() {
			super();
		}
		public void setmethodName(String mName) {
			methodName = mName;
		}
		public String getmethodName() {
			return methodName;
		}
		public void setreturnType(String mreturnType) {
			returnType = mreturnType;
		}
		public String getreturnType() {
			return returnType;
		}
		public void setparameterList(Hashtable pList) {
			parameterList = pList;
		}
		public Hashtable getparameterList() {
			return parameterList;
```

```
        }
        public void setnumberParameters(int n) {
                numberParameters = n;
        }
        public int getnumberParameters() {
                return numberParameters;
        }
}
```

Parameter.java

```
/** This class is used to store the details of the parameters of methods of components
 *  after parsing the UMM specification XML document
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.io.*;
import java.lang.*;
import java.util.*;

public class Parameter {

        String parameterName;
        String returnType;

        public Parameter() {
                super();
        }
        public void setparameterName(String pName) {
                parameterName = pName;
        }
        public String getparameterName() {
                return parameterName;
        }
        public void setreturnType(String preturnType) {
                returnType = preturnType;
        }
        public String getreturnType() {
                return returnType;
        }
}
```

Graphical User Interface Components

GlueGeneratorGUI.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

/**
 * This class provides a graphical user inerface for the GlueGenerator System
 * to accept system name to be composed.
 *
 * @author Kalpana Tummala
 * @date April 2004
 * @version 1.0
 */

class GlueGeneratorGUI extends JFrame
{
        private static ORB orb;
        private static org.omg.CosNaming.NamingContext root_ctx;

        GlueGeneratorKB GlueGeneratorKB;
  JTextField systemNameField;
  JTextArea systemDetailsArea;
  JButton enter_button;
  JButton exit_button;

  public GlueGeneratorGUI()
  {
    Box holdingBox = new Box(BoxLayout.Y_AXIS);

    JPanel facePanel = new JPanel();
    //facePanel.setLayout(new GridLayout(2, 1));
    JTextField faceField = new JTextField("Glue Generator System");
    faceField.setEditable(false);
    faceField.setFont(new Font("Serif", Font.ITALIC + Font.BOLD, 20));
    faceField.setHorizontalAlignment(JTextField.CENTER);
    facePanel.add(faceField);

    JPanel systemNameLabelPanel = new JPanel();
    JLabel systemNameLabel = new JLabel("System Name: ");
    systemNameLabel.setHorizontalAlignment(SwingConstants.RIGHT);
    systemNameLabelPanel.add(systemNameLabel);
    JPanel systemNamePanel = new JPanel();
    systemNameField = new JTextField(20);
    systemNamePanel.add(systemNameField);

    JPanel connectedToLabelPanel = new JPanel();
    JLabel connectedToLabel = new JLabel("System Details:");
    connectedToLabelPanel.add(connectedToLabel);
    JPanel connectedToPanel = new JPanel();
    systemDetailsArea = new JTextArea();
    systemDetailsArea.setColumns(20);
    systemDetailsArea.setRows(2);
    systemDetailsArea.setEditable(false);
    connectedToPanel.add(systemDetailsArea);

    Listener listener = new Listener();
    enter_button = new JButton("Enter");
    enter_button.addActionListener(listener);
    exit_button = new JButton("Exit");
    exit_button.addActionListener(listener);
    JPanel button_panel = new JPanel();
    button_panel.add(enter_button);
    button_panel.add(exit_button);
```

```
        holdingBox.add(facePanel);
        holdingBox.add(systemNameLabelPanel);
        holdingBox.add(systemNamePanel);
                holdingBox.add(connectedToLabelPanel);
                holdingBox.add(connectedToPanel);
        holdingBox.add(button_panel);
        java.awt.Container content = getContentPane();
        content.add(holdingBox, "Center");

        addWindowListener(new WindowCloser());
        setLocation(20, 20);
        setSize(400, 300);
        setResizable(false);
    }

    private class Listener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            java.lang.Object source = e.getSource();
            try
            {
                if(source == enter_button)
                {
                    String system_name = systemNameField.getText();

                    if(system_name != null && !system_name.equals(""))
                    {
                        systemNameField.setText(system_name);
                        systemNameField.repaint();
                        BufferedWriter fileWriter = new BufferedWriter(new FileWriter(system_name + "_Spec.txt"));
                                        fileWriter.write("System Name: " + system_name + "\n");
                                        systemDetailsArea.setText("System Name: " + system_name + "\n" + "System
Version in KB: 1.0" + "\n" + "Details in KB last modified: May 10 2004 " + "\n");
                                        fileWriter.close();
                        systemDetailsArea.repaint();

                                        GlueGeneratorKB = new GlueGeneratorKB();
                                        if(GlueGeneratorKB.validSystemName(system_name).equals("valid")){
                                                GlueGeneratorKB.setSystemName(system_name);
                                                new GlueCompGUI(GlueGeneratorKB, orb,
root_ctx).setVisible(true);
                                        }
                                        else{
                                                JOptionPane.showMessageDialog(null, "SystemName INVALID!");
                                        }
                    }
                                else
                                        JOptionPane.showMessageDialog(null, "Enter SystemName!");
                }
                else if(source == exit_button)
                {
                    System.exit(0);
                }
            }
            catch(Exception ex)
            {}
        }
    }
    private class WindowCloser extends WindowAdapter
    {
        public void windowClosing(WindowEvent event)
        {
           System.exit(0);
        }
    }

            public static void main(String args[]){
                        // initialize ORB and locate object
                        try {
                                    // initialize ORB
                                    orb = ORB.init(args, null);
                                    // Initialize object reference for Naming Service
                                    root_ctx =
org.omg.CosNaming.NamingContextHelper.narrow(orb.resolve_initial_references("NameService"));
                                    System.out.println("Initialized ORB");
                        } catch (Exception e) {
                                    System.err.println ("Exception initializing ORB:" + e);
                                    return;
                        }
                                    new GlueGeneratorGUI().setVisible(true);
            }
}
```

GlueCompGUI.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.io.*;

/**
 * This class provides a graphical user inerface for the GlueGenerator System
 * to accept component details of the system to be composed.
 *
 * @author Kalpana Tummala
 * @date April 2004
 * @version 1.0
 */

class GlueCompGUI extends JFrame
{
            ORB orb;
            org.omg.CosNaming.NamingContext root_ctx;

            String SystemName;
            int numberComponents;
            String component_name;
            String UMMSpecification_file;
            JTextField componentNameField;
            JTextField UMMSpecificationNameField;
            JTextArea componentDetailsArea;
            JButton enter_button;
            JButton exit_button;
            GlueGeneratorKB GlueGeneratorKB;
            Hashtable ummSpecTable;
            Hashtable umm_Spec_Table;
	Hashtable compSpec;
	int num = 1;
	String buf = "";
	BufferedWriter fileWriter;

	public GlueCompGUI(GlueGeneratorKB GlueGeneratorKB, ORB orb, org.omg.CosNaming.NamingContext
root_ctx)
	{
		this.GlueGeneratorKB = GlueGeneratorKB;
		this.orb = orb;
		this.root_ctx = root_ctx;
		this.SystemName = GlueGeneratorKB.getSystemName();
		this.numberComponents = GlueGeneratorKB.getnumberComponents();
		this.ummSpecTable = GlueGeneratorKB.getummSpecTable();
		this.umm_Spec_Table = GlueGeneratorKB.getummSpecTable();
		compSpec = new Hashtable();

		Box holdingBox = new Box(BoxLayout.Y_AXIS);

		JPanel facePanel = new JPanel();
		//facePanel.setLayout(new GridLayout(2, 1));
		JTextField faceField = new JTextField("Glue Generator System for " + SystemName + " System");
		faceField.setEditable(false);
		faceField.setFont(new Font("Serif", Font.ITALIC + Font.BOLD, 20));
		faceField.setHorizontalAlignment(JTextField.CENTER);
		facePanel.add(faceField);

		JPanel componentNameLabelPanel = new JPanel();
		JLabel componentNameLabel = new JLabel("Component Name: ");
		componentNameLabel.setHorizontalAlignment(SwingConstants.RIGHT);
		componentNameLabelPanel.add(componentNameLabel);
		JPanel componentNamePanel = new JPanel();
		componentNameField = new JTextField(20);
		componentNamePanel.add(componentNameField);

		JPanel UMMSpecificationNameLabelPanel = new JPanel();
		JLabel UMMSpecificationNameLabel = new JLabel("UMMSpecification File: ");
		UMMSpecificationNameLabel.setHorizontalAlignment(SwingConstants.RIGHT);
		UMMSpecificationNameLabelPanel.add(UMMSpecificationNameLabel);
		JPanel UMMSpecificationNamePanel = new JPanel();
		UMMSpecificationNameField = new JTextField(50);
		UMMSpecificationNamePanel.add(UMMSpecificationNameField);

		JPanel connectedToLabelPanel = new JPanel();
		JLabel connectedToLabel = new JLabel("Component Details:");
		connectedToLabelPanel.add(connectedToLabel);
		JPanel connectedToPanel = new JPanel();
		componentDetailsArea = new JTextArea();
		componentDetailsArea.setColumns(20);
		componentDetailsArea.setRows(2);
		componentDetailsArea.setEditable(false);
		connectedToPanel.add(componentDetailsArea);

		Listener listener = new Listener();
		enter_button = new JButton("Enter");
		enter_button.addActionListener(listener);
		exit_button = new JButton("Exit");
		exit_button.addActionListener(listener);
		JPanel button_panel = new JPanel();
		button_panel.add(enter_button);
		button_panel.add(exit_button);
```

```
				holdingBox.add(facePanel);
				holdingBox.add(componentNameLabelPanel);
				holdingBox.add(componentNamePanel);
				holdingBox.add(UMMSpecificationNameLabelPanel);
				holdingBox.add(UMMSpecificationNamePanel);
				holdingBox.add(connectedToLabelPanel);
				holdingBox.add(connectedToPanel);
				holdingBox.add(button_panel);
				java.awt.Container content = getContentPane();
				content.add(holdingBox, "Center");

				addWindowListener(new WindowCloser());
				setLocation(20, 20);
				setSize(600, 400);
				setResizable(true);
				try{
					fileWriter = new BufferedWriter(new FileWriter(SystemName + "_Spec.txt", true));
				}catch(IOException ioe){
				}
		}

		private class Listener implements ActionListener
		{
				public void actionPerformed(ActionEvent e)
				{
					java.lang.Object source = e.getSource();
					try
					{
						if(source == enter_button)
						{
							if(num <= numberComponents){
								String component_name =
componentNameField.getText();
								String UMMSpecification_file =
UMMSpecificationNameField.getText();
								if(component_name != null &&
!component_name.equals("") &&
								UMMSpecification_file != null &&
!UMMSpecification_file.equals(""))
								{
									componentNameField.setText("");
									componentNameField.repaint();
									UMMSpecificationNameField.setText("");
									UMMSpecificationNameField.repaint();

									Enumeration enum = ummSpecTable.keys();
									String key;
									String value;
									String validity = "false";
									while (enum.hasMoreElements()) {
										key = (String) enum.nextElement();
										value = (String)
ummSpecTable.get(key);
										if(component_name.equals(key) &&
UMMSpecification_file.equals(value)){
											ummSpecTable.remove(key);
											validity = "true";
											break;
										}
									}
									if(validity.equals("true")){
										buf = buf + component_name + " - " +
UMMSpecification_file + "\n";
                                                                        fileWriter.write("C" + num + "-->" +
"UMM" + num + "\n");
                                                                        fileWriter.write("C" + num + ": " +
component_name + "\n");
                                                                        compSpec.put("C" + num,
component_name);
                                                                        fileWriter.write("UMM" + num + ": " +
UMMSpecification_file + "\n");
                                                                        num++;
                                                                        componentDetailsArea.setText(buf);
                                                                        componentDetailsArea.repaint();
                                                                        if(num > numberComponents){

        JOptionPane.showMessageDialog(null, "All Component details have been obtained");
        GlueGeneratorKB.setummSpecTable(umm_Spec_Table);
        fileWriter.close();
        new GlueCompInteractionsGUI(GlueGeneratorKB, orb, root_ctx, compSpec).setVisible(true);
                                                                        }
                                                                }else{
                                                                        JOptionPane.showMessageDialog(null,
"Component Name: " + component_name + " and/or UMMSpecFile: " + UMMSpecification_file + " are INvalid!");
                                                                        componentDetailsArea.setText(buf);
                                                                        componentDetailsArea.repaint();
                                                                }
                                                        }
                                                }
                                                else
                                                        JOptionPane.showMessageDialog(null, "All Components
required for the " + SystemName + " have been already obtained !");
                                        }
                                        else if(source == exit_button)
                                        {
                                                System.exit(0);
                                        }
                                }
                                catch(Exception ex)
                                {}
                }
        }
        private class WindowCloser extends WindowAdapter
        {
                public void windowClosing(WindowEvent event)
                {
                        System.exit(0);
                }
        }
}
```

GlueCompInteractionsGUI.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.io.*;
import java.util.*;
```

```
/**
 * This class provides a graphical user inerface for the GlueGenerator System
 * to accept component interactions of the system to be composed.
 *
 * @author Kalpana Tummala
 * @date April 2004
 * @version 1.0
 */

class GlueCompInteractionsGUI extends JFrame
{
        ORB orb;
        org.omg.CosNaming.NamingContext root_ctx;

        String SystemName;
        String ServerOption;
        int numberInteractions;
        String initiator;
        String responder;
        JTextField initiatorNameField;
        JTextField responderNameField;
        JTextArea componentDetailsArea;
        JButton enter_button;
        JButton exit_button;
        GlueGeneratorKB GlueGeneratorKB;
        String[][] ComponentInteractionTable;
        String[][] Component_Interaction_Table;
        int num = 1;
        int s = 0;
        String buf = "";
        String cl,ser;
        BufferedWriter fileWriter;
        Hashtable compSpec;

        public GlueCompInteractionsGUI(GlueGeneratorKB GlueGeneratorKB, ORB orb,
org.omg.CosNaming.NamingContext root_ctx, Hashtable compSpec)
        {
                this.GlueGeneratorKB = GlueGeneratorKB;
                this.orb = orb;
                this.root_ctx = root_ctx;
                this.SystemName = GlueGeneratorKB.getSystemName();
                this.ServerOption = GlueGeneratorKB.getServerOption();
                this.numberInteractions = GlueGeneratorKB.getnumberInteractions();
                this.ComponentInteractionTable = GlueGeneratorKB.getComponentInteractionTable();
                this.Component_Interaction_Table = GlueGeneratorKB.getComponentInteractionTable();
                this.compSpec = compSpec;

                Box holdingBox = new Box(BoxLayout.Y_AXIS);

                JPanel facePanel = new JPanel();
                //facePanel.setLayout(new GridLayout(2, 1));
                JTextField faceField = new JTextField("Glue Generator System for " + SystemName + " System");
                faceField.setEditable(false);
                faceField.setFont(new Font("Serif", Font.ITALIC + Font.BOLD, 20));
                faceField.setHorizontalAlignment(JTextField.CENTER);
                facePanel.add(faceField);

                JPanel initiatorNameLabelPanel = new JPanel();
                JLabel initiatorNameLabel = new JLabel("Initiator Name: ");
                initiatorNameLabel.setHorizontalAlignment(SwingConstants.RIGHT);
                initiatorNameLabelPanel.add(initiatorNameLabel);
                JPanel initiatorNamePanel = new JPanel();
```

```
		initiatorNameField = new JTextField(20);
		initiatorNamePanel.add(initiatorNameField);

		JPanel responderNameLabelPanel = new JPanel();
		JLabel responderNameLabel = new JLabel("Responder Name: ");
		responderNameLabel.setHorizontalAlignment(SwingConstants.RIGHT);
		responderNameLabelPanel.add(responderNameLabel);
		JPanel responderNamePanel = new JPanel();
		responderNameField = new JTextField(20);
		responderNamePanel.add(responderNameField);

		JPanel connectedToLabelPanel = new JPanel();
		JLabel connectedToLabel = new JLabel("Component Interaction Details:");
		connectedToLabelPanel.add(connectedToLabel);
		JPanel connectedToPanel = new JPanel();
		componentDetailsArea = new JTextArea();
		componentDetailsArea.setColumns(20);
		componentDetailsArea.setRows(2);
		componentDetailsArea.setEditable(false);
		connectedToPanel.add(componentDetailsArea);

		Listener listener = new Listener();
		enter_button = new JButton("Enter");
		enter_button.addActionListener(listener);
		exit_button = new JButton("Exit");
		exit_button.addActionListener(listener);
		JPanel button_panel = new JPanel();
		button_panel.add(enter_button);
		button_panel.add(exit_button);

		holdingBox.add(facePanel);
		holdingBox.add(initiatorNameLabelPanel);
		holdingBox.add(initiatorNamePanel);
		holdingBox.add(responderNameLabelPanel);
		holdingBox.add(responderNamePanel);
		holdingBox.add(connectedToLabelPanel);
		holdingBox.add(connectedToPanel);
		holdingBox.add(button_panel);
		java.awt.Container content = getContentPane();
		content.add(holdingBox, "Center");

		addWindowListener(new WindowCloser());
		setLocation(20, 20);
		setSize(400, 400);
		setResizable(true);
		try{
			fileWriter = new BufferedWriter(new FileWriter(SystemName + "_Spec.txt", true));
			fileWriter.write("System-->");
		}catch(IOException ioe){}
	}

	private class Listener implements ActionListener
	{
		public void actionPerformed(ActionEvent e)
		{
			java.lang.Object source = e.getSource();
			try
			{
				if(source == enter_button)
				{
					String validity = "false";
					if(num <= numberInteractions){
```

```
                                                            String initiator = initiatorNameField.getText();
                                                            String responder = responderNameField.getText();

                                                            if(initiator != null && !initiator.equals("") &&
                                                            responder != null && !responder.equals(""))
                                                            {
                                                                    initiatorNameField.setText("");
                                                                    initiatorNameField.repaint();
                                                                    responderNameField.setText("");
                                                                    responderNameField.repaint();
                                                                    String initiator_responder_pair = initiator + "-->"
+ responder;
                                                                    for (int i = 0; i < numberInteractions; i++){

        if(ComponentInteractionTable[i][0].equals(SystemName)){
                String init = ComponentInteractionTable[i][1];
                String resp = ComponentInteractionTable[i][2];
                String init_resp_pair = init + "-->" + resp;

        if(initiator_responder_pair.equals(init_resp_pair)){
        ComponentInteractionTable[i][1] = "";
        ComponentInteractionTable[i][2] = "";
        validity = "true";
        break;
        }
        }
        }
                if(validity.equals("true")){
                        num++;
                        buf = buf + initiator + "-->" + responder + "\n";
                        Enumeration enum = compSpec.keys();
                        String key;
                        String value;
                        while (enum.hasMoreElements()) {
                                key = (String) enum.nextElement();
                                value = (String) compSpec.get(key);
                                if(initiator.equals(value)){
                                        cl = key;
                                }else if(responder.equals(value)){
                                ser = key;
                        }
                }
                if(num == 2){
                        fileWriter.write(cl + "comm" + ser);
                }
                else
                        fileWriter.write(";" + cl + "comm" + ser);
                        componentDetailsArea.setText(buf);
                        componentDetailsArea.repaint();
                        if(num > numberInteractions){
                                JOptionPane.showMessageDialog(null, "All Component Interaction details
have been obtained!");

        GlueGeneratorKB.setComponentInteractionTable(Component_Interaction_Table);
                fileWriter.close();
                new SysArchGUI(GlueGeneratorKB, orb, root_ctx).setVisible(true);
                                                                            }
                                                                    }else{
                                                                            JOptionPane.showMessageDialog(null,
"Initiator Name: " + initiator + " and Responder Name: " + responder + " are INVALID!");
                                                                            componentDetailsArea.setText(buf);
                                                                            componentDetailsArea.repaint();
								}

							}
							else
								JOptionPane.showMessageDialog(null, 
"INVALID INPUT");
						}
						else
							JOptionPane.showMessageDialog(null, "All Component 
Interaction details have been already obtained !");
					}
					else if(source == exit_button)
					{
						System.exit(0);
					}
				}
				catch(Exception ex)
				{}
			}
		}
		private class WindowCloser extends WindowAdapter
		{
			public void windowClosing(WindowEvent event)
			{
				System.exit(0);
			}
		}
}
```

SysArchGUI.java

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.text.NumberFormat;
import java.io.*;
import java.rmi.*;
import java.util.*;
import java.io.*;
import java.rmi.server.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

/**
 * This class provides a graphical user interface for the GlueGenerator System
 * to show System architecture and start glue generation.
 *
 * @author Kalpana Tummala
 * @date April 2004
 * @version 1.0
 */

public class SysArchGUI extends JFrame
{
		ORB orb;
		org.omg.CosNaming.NamingContext root_ctx;

   private JComboBox PlacementofGlueSelectionBox;
   private JButton ShowSysArchButton;
   private JTextArea ArchDetailsArea;
   private JButton SystemGenButton;
private Box bigBox;
private JScrollPane scrollPane;
String SystemName;
GlueGeneratorKB GlueGeneratorKB;
        BufferedReader fileReader;
        BufferedWriter fileWriter;
        String[][] ComponentInteractionTable;
        //ComponentInteractionTable: SystemName(0) - Inititaor(1) - Responder(2) pairs
        Hashtable ummSpecTable;
        int numberInteractions;
        String techbuf = "";

/**
 * Constructor. Set up the graphical user interface.
 */
public SysArchGUI(GlueGeneratorKB GlueGeneratorKB, ORB orb, org.omg.CosNaming.NamingContext root_ctx)
{
    this.GlueGeneratorKB = GlueGeneratorKB;
        this.orb = orb;
        this.root_ctx = root_ctx;
          this.SystemName = GlueGeneratorKB.getSystemName();

   Listener listener = new Listener();

                    JPanel facePanel = new JPanel();
                    //facePanel.setLayout(new GridLayout(2, 1));
                    JTextField faceField = new JTextField(SystemName + " System Architecture");
                    faceField.setEditable(false);
                    faceField.setFont(new Font("Serif", Font.ITALIC + Font.BOLD, 20));
                    faceField.setHorizontalAlignment(JTextField.CENTER);
                    facePanel.add(faceField);

   JPanel PlacementofGlueSelectionLabelPanel = new JPanel();
   JLabel PlacementofGlueSelectionLabel = new JLabel("Available Selections for Placement of Glue: ");
   PlacementofGlueSelectionLabel.setHorizontalAlignment(JTextField.CENTER);
   PlacementofGlueSelectionLabelPanel.add(PlacementofGlueSelectionLabel);

   JPanel PlacementofGlueSelectionPanel = new JPanel();
   PlacementofGlueSelectionBox = new JComboBox();
   PlacementofGlueSelectionBox.addItem("Centralized");
   PlacementofGlueSelectionBox.addItem("De-Centralized: Place Glue with Initiator");
   PlacementofGlueSelectionBox.addItem("De-Centralized: Place Glue with Responder");
   PlacementofGlueSelectionBox.addItem("De-Centralized: Place Glue randomly");
   PlacementofGlueSelectionPanel.add(PlacementofGlueSelectionBox);

   JPanel ShowSysArchButtonPanel = new JPanel();
   ShowSysArchButton = new JButton("ShowSystemArchitecture");
   ShowSysArchButton.addActionListener(listener);
   ShowSysArchButtonPanel.add(ShowSysArchButton);

          JPanel ArchDetailsLabelPanel = new JPanel();
          JLabel ArchDetailsLabel = new JLabel("System Architecture Details:");
          ArchDetailsLabelPanel.add(ArchDetailsLabel);

   JPanel ArchDetailsAreaPanel = new JPanel();
          ArchDetailsArea = new JTextArea();
          ArchDetailsArea.setColumns(40);
          ArchDetailsArea.setRows(10);
          ArchDetailsArea.setEditable(false);
          scrollPane = new JScrollPane(ArchDetailsArea);
          ArchDetailsAreaPanel.add(scrollPane);
```

```
                        JPanel SystemGenbuttonPanel = new JPanel();
      SystemGenButton = new JButton("SystemGeneration");
      SystemGenButton.addActionListener(listener);
      SystemGenbuttonPanel.add(SystemGenButton);

      bigBox = new Box(BoxLayout.Y_AXIS);
      bigBox.add(facePanel);
            bigBox.add(PlacementofGlueSelectionLabelPanel);
      bigBox.add(PlacementofGlueSelectionPanel);
                        bigBox.add(ShowSysArchButtonPanel);
                        bigBox.add(ArchDetailsLabelPanel);
                        bigBox.add(ArchDetailsAreaPanel);
      bigBox.add(SystemGenbuttonPanel);

      java.awt.Container contentPane = getContentPane();
      contentPane.add(bigBox, "Center");
      setSize(600, 800);
      setResizable(true);
      addWindowListener(new WindowCloser());
            try{
                        fileWriter = new BufferedWriter(new FileWriter(SystemName + "_Spec.txt", true));
                        fileReader = new BufferedReader(new FileReader(SystemName + "_Spec.txt"));
            }catch(IOException ioe){}
            ComponentInteractionTable = GlueGeneratorKB.getComponentInteractionTable();
                        ummSpecTable = GlueGeneratorKB.getummSpecTable();
                        numberInteractions = GlueGeneratorKB.getnumberInteractions();
   }

   /**
    * This private class implements the event listener.
    */
   private class Listener implements ActionListener
   {
      public void actionPerformed(ActionEvent event)
      {
         java.lang.Object source = event.getSource();

         try
         {   if(source == ShowSysArchButton)
            {
                                    String initiator = "";
                                    String initiatorUMMspec = "";
                                    String initiatorTech = "";
                                    String responder = "";
                                    String responderUMMspec = "";
                                    String responderTech = "";
                                    String init_resp_pair = "";

                                    techbuf = techbuf + "These are the technology differences present for Component
Interactions in the System: " + "\n";
                                    for (int i = 0; i < numberInteractions; i++){
                                                if(ComponentInteractionTable[i][0].equals(SystemName)){
                                                            initiator = ComponentInteractionTable[i][1];
                                                            initiatorUMMspec = (String)ummSpecTable.get(initiator);
                                                            UMMSpecificationParser kpinitiator = new
UMMSpecificationParser(initiatorUMMspec);
                                                            Component initiatorComp = new Component();
                                                            initiatorComp = kpinitiator.getComponent();
                                                            initiatorTech = initiatorComp.getcomponentTechnology();

                                                            responder = ComponentInteractionTable[i][2];
```

```
                                        responderUMMspec = (String)ummSpecTable.get(responder);
                                        UMMSpecificationParser kpresponder = new
UMMSpecificationParser(responderUMMspec);
                                        Component responderComp = new Component();
                                        responderComp = kpresponder.getComponent();
                                        responderTech = responderComp.getcomponentTechnology();
                                        if(initiatorTech.equals("Java RMI") &&
responderTech.equals("CORBA ORB")){
                                                techbuf = techbuf + initiator + "-" + responder + ": " +
initiatorTech + "-" + responderTech + "\n";
                                        }else if(initiatorTech.equals("CORBA ORB") &&
responderTech.equals("Java RMI")){
                                                techbuf = techbuf + initiator + "-" + responder + ": " +
initiatorTech + "-" + responderTech + "\n";
                                        }
                                }
                        }
         String PlacementofGlue = ((String)PlacementofGlueSelectionBox.getSelectedItem()).trim();
                        try{
          if(PlacementofGlue.equals("Centralized"))
          {
            JOptionPane.showMessageDialog(null, "Showing architecture details of the " + SystemName + "
System to be composed!");
                                fileWriter.write("\n" + "GluePlacement: " + PlacementofGlue);
                                fileWriter.close();
                                String buf = "";
                                String arch;
                                try{
                                        while((arch = fileReader.readLine()) != null){
                                                buf = buf + arch + "\n";
                                        }
                                        ArchDetailsArea.setText(buf);
                                        fileReader.close();
                                }catch(IOException ioe){}
                                techbuf = techbuf + "\n" + "To start generating the required glue for these
heterogeneous interactions, Click on SystemGeneration button!";
                                JOptionPane.showMessageDialog(null, techbuf);
          }
         else if(PlacementofGlue.equals("De-Centralized: Place Glue with Initiator"))
                        {
            JOptionPane.showMessageDialog(null, "Showing architecture details of the " + SystemName + "
System to be composed!");
                                fileWriter.write("\n" + "GluePlacement: " + PlacementofGlue);
                                fileWriter.close();
                                String buf = "";
                                String arch;
                                try{
                                        while((arch = fileReader.readLine()) != null){
                                                buf = buf + arch + "\n";
                                        }
                                        ArchDetailsArea.setText(buf);
                                        fileReader.close();
                                }catch(IOException ioe){}
                                techbuf = techbuf + "\n" + "To start generating the required glue for these
heterogeneous interactions, Click on SystemGeneration button!";
                                JOptionPane.showMessageDialog(null, techbuf);
          }
                        else if(PlacementofGlue.equals("De-Centralized: Place Glue with Responder"))
                        {
            JOptionPane.showMessageDialog(null, "Showing architecture details of the " + SystemName + "
System to be composed!");
                                fileWriter.write("\n" + "GluePlacement: " + PlacementofGlue);
```

```
                                        fileWriter.close();
                                        String buf = "";
                                        String arch;
                                        try{
                                                while((arch = fileReader.readLine()) != null){
                                                        buf = buf + arch + "\n";
                                                }
                                                ArchDetailsArea.setText(buf);
                                                fileReader.close();
                                        }catch(IOException ioe){}
                                        techbuf = techbuf + "\n" + "To start generating the required glue for these
heterogeneous interactions, Click on SystemGeneration button!";
                                        JOptionPane.showMessageDialog(null, techbuf);
          }

         else if(PlacementofGlue.equals("De-Centralized: Place Glue randomly"))
                                {
            JOptionPane.showMessageDialog(null, "Showing architecture details of the " + SystemName + "
System to be composed!");
                                        fileWriter.write("\n" + "GluePlacement: " + PlacementofGlue);
                                        fileWriter.close();
                                        String buf = "";
                                        String arch;
                                        try{
                                                while((arch = fileReader.readLine()) != null){
                                                        buf = buf + arch + "\n";
                                                }
                                                ArchDetailsArea.setText(buf);
                                                fileReader.close();
                                        }catch(IOException ioe){}
                                        techbuf = techbuf + "\n" + "To start generating the required glue for these
heterogeneous interactions, Click on SystemGeneration button!";
                                        JOptionPane.showMessageDialog(null, techbuf);
          }
                                }catch(IOException ioe){}

        }
        else if(source == SystemGenButton)
        {
                                        GlueGenerator GlueGen = new GlueGenerator(GlueGeneratorKB);
                                        GlueGen.startGG();
                                        new GlueQoSGUI(GlueGeneratorKB, orb, root_ctx).setVisible(true);
        }
/*        else if(source == exitButton)
        {
          System.exit(0);
        }
*/
             }
      catch(ClassCastException e)
      {
        JOptionPane.showMessageDialog(null, "Incorrect Input!");
      }
    }
  }

  /**
   * This private class implements the window closer event.
   */
  private class WindowCloser extends WindowAdapter
  {
    public void windowClosing(WindowEvent event)
```

```
        {
           System.exit(0);
        }
     }
}
```

GlueQoSGUI.java

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.text.NumberFormat;
import java.io.*;
import java.rmi.*;
import java.util.*;
import java.io.*;
import java.rmi.server.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

/**
 * This class provides a graphical user inerface for the dynamic QoS testing of the composed System.
 *
 * @author Kalpana Tummala
 * @date April 2004
 * @version 1.0
 */

public class GlueQoSGUI extends JFrame
{
		ORB orb;
		org.omg.CosNaming.NamingContext root_ctx;

   private JComboBox QoSTestSelectionBox;
   private JButton QoSTestButton;
   private JTextArea QoSValueDetailsArea;
   private Box bigBox;
   private JScrollPane scrollPane;
   String SystemName;
GlueGeneratorKB GlueGeneratorKB;
   String buf = "";

   /**
    * Constructor. Set up the graphical user interface.
    */
   public GlueQoSGUI(GlueGeneratorKB GlueGeneratorKB, ORB orb, org.omg.CosNaming.NamingContext
root_ctx)
   {
this.GlueGeneratorKB = GlueGeneratorKB;
		this.orb = orb;
		this.root_ctx = root_ctx;
		  this.SystemName = GlueGeneratorKB.getSystemName();

       Listener listener = new Listener();

				JPanel facePanel = new JPanel();
				//facePanel.setLayout(new GridLayout(2, 1));
				JTextField faceField = new JTextField("Glue QoS System for " + SystemName + " System");
				faceField.setEditable(false);
				faceField.setFont(new Font("Serif", Font.ITALIC + Font.BOLD, 20));
				faceField.setHorizontalAlignment(JTextField.CENTER);
```

```
                    facePanel.add(faceField);

    JPanel QoSTestSelectionLabelPanel = new JPanel();
    JLabel QoSTestSelectionLabel = new JLabel("Available Selections for QoSTest: ");
    QoSTestSelectionLabel.setHorizontalAlignment(JTextField.CENTER);
    QoSTestSelectionLabelPanel.add(QoSTestSelectionLabel);

    JPanel QoSTestSelectionPanel = new JPanel();
    QoSTestSelectionBox = new JComboBox();
    QoSTestSelectionBox.addItem("CTBank");
    QoSTestSelectionBox.addItem("CTValidation");
    QoSTestSelectionBox.addItem("ATMBank");
    QoSTestSelectionBox.addItem("ATMValidation");
    QoSTestSelectionPanel.add(QoSTestSelectionBox);

    JPanel QoSTestbuttonPanel = new JPanel();
    QoSTestButton = new JButton("QoSTest");
    QoSTestButton.addActionListener(listener);
    QoSTestbuttonPanel.add(QoSTestButton);

            JPanel QoSValuesLabelPanel = new JPanel();
            JLabel QoSValuesLabel = new JLabel("Component QoS Details:");
            QoSValuesLabelPanel.add(QoSValuesLabel);

    JPanel QoSValueDetailsAreaPanel = new JPanel();
            QoSValueDetailsArea = new JTextArea();
            QoSValueDetailsArea.setColumns(40);
            QoSValueDetailsArea.setRows(10);
            QoSValueDetailsArea.setEditable(false);
            scrollPane = new JScrollPane(QoSValueDetailsArea);
            QoSValueDetailsAreaPanel.add(scrollPane);

    bigBox = new Box(BoxLayout.Y_AXIS);
    bigBox.add(facePanel);
    bigBox.add(QoSTestSelectionLabelPanel);
    bigBox.add(QoSTestSelectionPanel);
    bigBox.add(QoSTestbuttonPanel);
    bigBox.add(QoSValuesLabelPanel);
    bigBox.add(QoSValueDetailsAreaPanel);

    java.awt.Container contentPane = getContentPane();
    contentPane.add(bigBox, "Center");
    setSize(600, 800);
    setResizable(true);
    addWindowListener(new WindowCloser());
  }

  /**
   * This private class implements the event listener.
   */
  private class Listener implements ActionListener
  {
    public void actionPerformed(ActionEvent event)
    {
      java.lang.Object source = event.getSource();

      try
      {
                          if(source == QoSTestButton)
        {
          String QoSTest = ((String)QoSTestSelectionBox.getSelectedItem()).trim();
          if(QoSTest.equals("CTBank"))
		{
						SystemCTTest CTTest = new SystemCTTest();
						CTTest.system_dynamic_test_Bank();
						double CTBank_endToEndDelay = CTTest.CTBank_endToEndDelay;
						int CTBank_throughput = CTTest.CTBank_throughput;
						buf = buf + "CashierTerminal Bank endToEndDelay: " +
CTBank_endToEndDelay + "\n" + "CashierTerminal Bank throughput: " + CTBank_throughput + "\n";
						QoSValueDetailsArea.setText(buf);
						QoSValueDetailsArea.repaint();

		}else if(QoSTest.equals("CTValidation"))
		{
						SystemCTTest CTTest = new SystemCTTest();
						CTTest.system_dynamic_test_Val();
						double CTVal_endToEndDelay = CTTest.CTVal_endToEndDelay;
						int CTVal_throughput = CTTest.CTVal_throughput;
						buf = buf + "CashierTerminal Val endToEndDelay: " +
CTVal_endToEndDelay + "\n" + "CashierTerminal Val throughput: " + CTVal_throughput + "\n";
						QoSValueDetailsArea.setText(buf);
						QoSValueDetailsArea.repaint();

						SystemCTGlueTest CTGlueTest = new SystemCTGlueTest();
						CTGlueTest.system_dynamic_test_Val();
						double CTGlueVal_endToEndDelay =
CTGlueTest.CTGlueVal_endToEndDelay;
						int CTGlueVal_throughput = CTGlueTest.CTGlueVal_throughput;
						buf = buf + "CashierTerminal Val Glue endToEndDelay: " +
CTGlueVal_endToEndDelay + "\n" + "CashierTerminal Val Glue throughput: " + CTGlueVal_throughput + "\n";
						QoSValueDetailsArea.setText(buf);
						QoSValueDetailsArea.repaint();

						double Glueoverhead = CTVal_endToEndDelay -
CTGlueVal_endToEndDelay;
						buf = buf + "Glue Overhead for CashierTerminal Validation: " + Glueoverhead
+ "\n";
						QoSValueDetailsArea.setText(buf);
						QoSValueDetailsArea.repaint();

		}else if(QoSTest.equals("ATMBank"))
		{
						SystemATMTest ATMTest = new SystemATMTest(orb, root_ctx);
						ATMTest.system_dynamic_test_Bank();
						double ATMBank_endToEndDelay = ATMTest.ATMBank_endToEndDelay;
						int ATMBank_throughput = ATMTest.ATMBank_throughput;
						buf = buf + "ATM Bank endToEndDelay: " + ATMBank_endToEndDelay +
"\n" + "ATM Bank throughput: " + ATMBank_throughput + "\n";
						QoSValueDetailsArea.setText(buf);
						QoSValueDetailsArea.repaint();

						SystemATMGlueTest ATMGlueTest = new SystemATMGlueTest(orb,
root_ctx);
						ATMGlueTest.system_dynamic_test_Bank();
						double ATMGlueBank_endToEndDelay =
ATMGlueTest.ATMGlueBank_endToEndDelay;
						int ATMGlueBank_throughput = ATMGlueTest.ATMGlueBank_throughput;
						buf = buf + "ATM Bank Glue endToEndDelay: " +
ATMGlueBank_endToEndDelay + "\n" + "ATM Bank Glue throughput: " + ATMGlueBank_throughput + "\n";
						QoSValueDetailsArea.setText(buf);
						QoSValueDetailsArea.repaint();

						double Glueoverhead = ATMBank_endToEndDelay -
ATMGlueBank_endToEndDelay;
```

```
                                        buf = buf + "Glue Overhead for ATM Bank: " + Glueoverhead + "\n";
                                        QoSValueDetailsArea.setText(buf);
                                        QoSValueDetailsArea.repaint();

                }else if(QoSTest.equals("ATMValidation"))
                {
                                        SystemATMTest ATMTest = new SystemATMTest(orb, root_ctx);
                                        ATMTest.system_dynamic_test_Val();
                                        double ATMVal_endToEndDelay = ATMTest.ATMVal_endToEndDelay;
                                        int ATMVal_throughput = ATMTest.ATMVal_throughput;
                                        buf = buf + "ATM Val endToEndDelay: " + ATMVal_endToEndDelay + "\n"
+ "ATM Val throughput: " + ATMVal_throughput + "\n";
                                        QoSValueDetailsArea.setText(buf);
                                        QoSValueDetailsArea.repaint();

                                        SystemATMGlueTest ATMGlueTest = new SystemATMGlueTest(orb,
root_ctx);
                                        ATMGlueTest.system_dynamic_test_Val();
                                        double ATMGlueVal_endToEndDelay =
ATMGlueTest.ATMGlueVal_endToEndDelay;
                                        int ATMGlueVal_throughput = ATMGlueTest.ATMGlueVal_throughput;
                                        buf = buf + "ATM Val Glue endToEndDelay: " +
ATMGlueVal_endToEndDelay + "\n" + "ATM Val Glue throughput: " + ATMGlueVal_throughput + "\n";
                                        QoSValueDetailsArea.setText(buf);
                                        QoSValueDetailsArea.repaint();

                                        double Glueoverhead = ATMVal_endToEndDelay -
ATMGlueVal_endToEndDelay;
                                        buf = buf + "Glue Overhead for ATM Validation: " + Glueoverhead + "\n";
                                        QoSValueDetailsArea.setText(buf);
                                        QoSValueDetailsArea.repaint();

                }
              }
/*            else if(source == exitButton)
              {
                System.exit(0);
              }
*/          }
          catch(ClassCastException e)
          {
            JOptionPane.showMessageDialog(null, "Incorrect Input!");
          }
      }
   }

   /**
    * This private class implements the window closer event.
    */
   private class WindowCloser extends WindowAdapter
   {
      public void windowClosing(WindowEvent event)
      {
         System.exit(0);
      }
   }
}
```

QoS Testing Components

SystemCTTest.java

```
import java.rmi.*;

public class SystemCTTest{

        public double CTBank_endToEndDelay = 0.0;
        public int CTBank_throughput = 0;
        public double CTVal_endToEndDelay = 0.0;
        public int CTVal_throughput = 0;

        public SystemCTTest()
        {
        }

        public void system_dynamic_test_Bank()
        {
                try
        {
                        IAccountManagement CTAcc =
(IAccountManagement)Naming.lookup("//134.68.140.101:9000/CashierTerminal");
                        ICustomerManagement CTTra =
(ICustomerManagement)Naming.lookup("//134.68.140.101:9000/CashierTerminal");
                BankQoSTesting bankQoSTesting = new BankQoSTesting();
                SystemQoS systemQoS = null;

        for (int i = 0; i < 50; i++)
        {
                        bankQoSTesting.startTimer("openAccount");
                       AccountInfo accountInfo_o1 = CTTra.openAccount("test1", "11111", 1);
            System.out.println("Account Opened Successfully\nAccount number: " +
accountInfo_o1.accountNumber +
                            "\nAccount type: " + ((accountInfo_o1.accountType == 1) ? "Saving Account" :
"Checking Account"));
                        bankQoSTesting.stopTimer("openAccount");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("openAccount"));

                        bankQoSTesting.startTimer("openAccount");
                        AccountInfo accountInfo_o0 = CTTra.openAccount("test2", "00000", 1);
            System.out.println("Account Opened Successfully\nAccount number: " +
accountInfo_o0.accountNumber +
                            "\nAccount type: " + ((accountInfo_o0.accountType == 1) ? "Saving Account" :
"Checking Account"));
                        bankQoSTesting.stopTimer("openAccount");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("openAccount"));

                        bankQoSTesting.startTimer("depositMoney");
                        CTAcc.depositMoney(100, "11111", 1);
                                System.out.println("deposit 11111");
                        bankQoSTesting.stopTimer("depositMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

                        bankQoSTesting.startTimer("depositMoney");
                        CTAcc.depositMoney(100, "00000", 1);
                                System.out.println("deposit 00000");
                        bankQoSTesting.stopTimer("depositMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

                        bankQoSTesting.startTimer("transferMoney");
                        CTAcc.transferMoney(50, "11111", 1, "00000", 1);
                                System.out.println("transfer 11111 to 00000");
                        bankQoSTesting.stopTimer("transferMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("transferMoney"));

                        bankQoSTesting.startTimer("transferMoney");
                        CTAcc.transferMoney(25, "00000", 1, "11111", 1);
                                System.out.println("transfer 00000 to 11111");
                        bankQoSTesting.stopTimer("transferMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("transferMoney"));

                        bankQoSTesting.startTimer("withdrawMoney");
                        CTAcc.withdrawMoney(75, "11111", 1);
                                System.out.println("withdraw 11111");
                        bankQoSTesting.stopTimer("withdrawMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

                        bankQoSTesting.startTimer("withdrawMoney");
                        CTAcc.withdrawMoney(125, "00000", 1);
                                System.out.println("withdraw 00000");
                        bankQoSTesting.stopTimer("withdrawMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

                        bankQoSTesting.startTimer("closeAccount");
                        AccountInfo accountInfo_c1 = CTTra.closeAccount("11111", 1);
                                System.out.println("Account Closed Successfully\nAccount number: " +
accountInfo_c1.accountNumber +
                            "\nAccount type: " + ((accountInfo_c1.accountType == 1) ? "Saving Account" :
"Checking Account"));
                        bankQoSTesting.stopTimer("closeAccount");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("closeAccount"));

                        bankQoSTesting.startTimer("closeAccount");
                        AccountInfo accountInfo_c0 = CTTra.closeAccount("00000", 1);
                                System.out.println("Account Closed Successfully\nAccount number: " +
accountInfo_c0.accountNumber +
                            "\nAccount type: " + ((accountInfo_c0.accountType == 1) ? "Saving Account" :
"Checking Account"));
                        bankQoSTesting.stopTimer("closeAccount");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("closeAccount"));
        }

        systemQoS = new SystemQoS("CTBank");

        //these are correct codes
        systemQoS.addSystemQoS("CTBank", "endToEndDelay", "" + bankQoSTesting.getEndToEndDelay());
        systemQoS.addSystemQoS("CTBank", "throughput", "" + bankQoSTesting.getThroughput());

        //these are temporary codes which adjust the number for demo purpose only

        //int throughput = bankQoSTesting.getThroughput() + 1000; //operations/second
        //double endToEndDelay = (1000000/throughput); //usecond

        //systemQoS.addSystemQoS("CTBank", "endToEndDelay", "" + endToEndDelay);
        //systemQoS.addSystemQoS("CTBank", "throughput", "" + throughput);

                        CTBank_endToEndDelay = bankQoSTesting.getEndToEndDelay();
                        CTBank_throughput = bankQoSTesting.getThroughput();
                System.out.println("CTBank endToEndDelay: " + CTBank_endToEndDelay);
                        System.out.println("CTBank throughput: " + CTBank_throughput);
        }
        catch(Exception e)
        {
        System.out.println(e);
        }

        }

        public void system_dynamic_test_Val()
        {
                try
        {
                        ICashierTerminal CT =
(ICashierTerminal)Naming.lookup("//134.68.140.101:9000/CashierTerminal");
                BankQoSTesting bankQoSTesting = new BankQoSTesting();
                SystemQoS systemQoS = null;

        for (int i = 0; i < 50; i++)
        {
                        bankQoSTesting.startTimer("validate");
                                boolean val = CT.validate("testid", "testpassword");
                                System.out.println("Validation of testid & testpassword : " + val);
                        bankQoSTesting.stopTimer("validate");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("validate"));
        }

        systemQoS = new SystemQoS("CTVal");

        //these are correct codes
        systemQoS.addSystemQoS("CTVal", "endToEndDelay", "" + bankQoSTesting.getEndToEndDelay());
        systemQoS.addSystemQoS("CTVal", "throughput", "" + bankQoSTesting.getThroughput());

        //these are temporary codes which adjust the number for demo purpose only

        //int throughput = bankQoSTesting.getThroughput() + 1000; //operations/second
        //double endToEndDelay = (1000000/throughput); //usecond

        //systemQoS.addSystemQoS("CTVal", "endToEndDelay", "" + endToEndDelay);
        //systemQoS.addSystemQoS("CTVal", "throughput", "" + throughput);

                        CTVal_endToEndDelay = bankQoSTesting.getEndToEndDelay();
                        CTVal_throughput = bankQoSTesting.getThroughput();
                        System.out.println("CTVal endToEndDelay: " + CTVal_endToEndDelay);
                        System.out.println("CTVal throughput: " + CTVal_throughput);
        }
        catch(Exception e)
        {
        System.out.println(e);
        }

  }
```

```
}
```

SystemCTGlueTest.java

```
import java.rmi.*;

public class SystemCTGlueTest{

		public double CTGlueVal_endToEndDelay = 0.0;
		public int CTGlueVal_throughput = 0;

		public SystemCTGlueTest()
		{
		}

		public void system_dynamic_test_Val()
		{
				try
		{
						ICashierValidationServer Glue_CS_RC_ICas =
(ICashierValidationServer)Naming.lookup("//134.68.140.101:9000/Glue_CS_RC_ICas");
				BankQoSTesting bankQoSTesting = new BankQoSTesting();
				SystemQoS systemQoS = null;

		for (int i = 0; i < 50; i++)
		{
						bankQoSTesting.startTimer("validate");
								boolean val = Glue_CS_RC_ICas.validate("testid", "testpassword");
								System.out.println("Validation of testid & testpassword : " + val);
						bankQoSTesting.stopTimer("validate");

		bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("validate"));
		}

		systemQoS = new SystemQoS("Glue_CS_RC_ICasVal");

		//these are correct codes
		systemQoS.addSystemQoS("Glue_CS_RC_ICasVal", "endToEndDelay", "" +
bankQoSTesting.getEndToEndDelay());
		systemQoS.addSystemQoS("Glue_CS_RC_ICasVal", "throughput", "" + bankQoSTesting.getThroughput());

		//these are temporary codes which adjust the number for demo purpose only

		//int throughput = bankQoSTesting.getThroughput() + 1000; //operations/second
		//double endToEndDelay = (1000000/throughput); //usecond

		//systemQoS.addSystemQoS("Glue_CS_RC_ICasVal", "endToEndDelay", "" + endToEndDelay);
		//systemQoS.addSystemQoS("Glue_CS_RC_ICasVal", "throughput", "" + throughput);

						CTGlueVal_endToEndDelay = bankQoSTesting.getEndToEndDelay();
						CTGlueVal_throughput = bankQoSTesting.getThroughput();
						System.out.println("CTGlueVal endToEndDelay: " + CTGlueVal_endToEndDelay);
						System.out.println("CTGlueVal throughput: " + CTGlueVal_throughput);
		}
		catch(Exception e)
		{
		System.out.println(e);
		}
   }

}
```

SystemATMTest.java

```
import java.util.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class SystemATMTest{

        private ORB orb;
        private org.omg.CosNaming.NamingContext root_ctx;
        private IATM ATM;

        public double ATMBank_endToEndDelay = 0.0;
        public int ATMBank_throughput = 0;
        public double ATMVal_endToEndDelay = 0.0;
        public int ATMVal_throughput = 0;

        public SystemATMTest(ORB orb, org.omg.CosNaming.NamingContext root_ctx)
        {
                this.orb = orb;
                this.root_ctx = root_ctx;
        }

        public void system_dynamic_test_Bank()
        {
                try {

                org.omg.CosNaming.NameComponent nc = new
org.omg.CosNaming.NameComponent("ATM","");
                org.omg.CosNaming.NameComponent[] name = {nc};
                org.omg.CORBA.Object obj = root_ctx.resolve(name);
                        ATM = IATMHelper.narrow(obj);
                } catch (org.omg.CosNaming.NamingContextPackage.NotFound nf) {
                        // this is ok;
                } catch (Exception e) {
                        System.out.println("Exception resolving name/binding new context");
                        System.out.println(e.toString());
                        System.exit(1);
                }

                try
        {
                        ICustomerManagement CT =
(ICustomerManagement)Naming.lookup("//134.68.140.101:9000/CashierTerminal");
                BankQoSTesting bankQoSTesting = new BankQoSTesting();
                SystemQoS systemQoS = null;

        for (int i = 0; i < 50; i++)
        {
                       AccountInfo accountInfo_o1 = CT.openAccount("test1", "11111", 1);
            System.out.println("Account Opened Successfully\nAccount number: " +
accountInfo_o1.accountNumber +
                           "\nAccount type: " + ((accountInfo_o1.accountType == 1) ? "Saving Account" :
"Checking Account"));

                        AccountInfo accountInfo_o0 = CT.openAccount("test2", "00000", 1);
            System.out.println("Account Opened Successfully\nAccount number: " +
accountInfo_o0.accountNumber +
                           "\nAccount type: " + ((accountInfo_o0.accountType == 1) ? "Saving Account" :
```

```
"Checking Account"));

                        bankQoSTesting.startTimer("depositMoney");
                        ATM.depositMoney(100, "11111", 1);
                                System.out.println("deposit 11111");
                        bankQoSTesting.stopTimer("depositMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

                        bankQoSTesting.startTimer("depositMoney");
                        ATM.depositMoney(100, "00000", 1);
                                System.out.println("deposit 00000");
                        bankQoSTesting.stopTimer("depositMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

                        bankQoSTesting.startTimer("withdrawMoney");
                        ATM.withdrawMoney(50, "11111", 1);
                                System.out.println("transferring: withdraw 11111");
                        bankQoSTesting.stopTimer("withdrawMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

                        bankQoSTesting.startTimer("depositMoney");
                        ATM.depositMoney(50, "00000", 1);
                                System.out.print(" to deposit 00000");
                        bankQoSTesting.stopTimer("depositMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

                        bankQoSTesting.startTimer("withdrawMoney");
                        ATM.withdrawMoney(25, "00000", 1);
                                System.out.println("transferring: withdraw 00000");
                        bankQoSTesting.stopTimer("withdrawMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

                        bankQoSTesting.startTimer("depositMoney");
                        ATM.depositMoney(25, "11111", 1);
                                System.out.print(" to deposit 11111");
                        bankQoSTesting.stopTimer("depositMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

                        bankQoSTesting.startTimer("withdrawMoney");
                        ATM.withdrawMoney(75, "11111", 1);
                                System.out.println("withdraw 11111");
                        bankQoSTesting.stopTimer("withdrawMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

                        bankQoSTesting.startTimer("withdrawMoney");
                        ATM.withdrawMoney(125, "00000", 1);
                                System.out.println("withdraw 00000");
                        bankQoSTesting.stopTimer("withdrawMoney");

        bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

                        AccountInfo accountInfo_c1 = CT.closeAccount("11111", 1);
                                System.out.println("Account Closed Successfully\nAccount number: " +
accountInfo_c1.accountNumber +
                              "\nAccount type: " + ((accountInfo_c1.accountType == 1) ? "Saving Account" :
```

```
"Checking Account"));

                                AccountInfo accountInfo_c0 = CT.closeAccount("00000", 1);
                                        System.out.println("Account Closed Successfully\nAccount number: " +
accountInfo_c0.accountNumber +
                                      "\nAccount type: " + ((accountInfo_c0.accountType == 1) ? "Saving Account" :
"Checking Account"));
        }

        systemQoS = new SystemQoS("ATMBank");

        //these are correct codes
        systemQoS.addSystemQoS("ATMBank", "endToEndDelay", "" + bankQoSTesting.getEndToEndDelay());
        systemQoS.addSystemQoS("ATMBank", "throughput", "" + bankQoSTesting.getThroughput());

        //these are temporary codes which adjust the number for demo purpose only

        //int throughput = bankQoSTesting.getThroughput() + 1000; //operations/second
        //double endToEndDelay = (1000000/throughput); //usecond

        //systemQoS.addSystemQoS("ATMBank", "endToEndDelay", "" + endToEndDelay);
        //systemQoS.addSystemQoS("ATMBank", "throughput", "" + throughput);

                                ATMBank_endToEndDelay = bankQoSTesting.getEndToEndDelay();
                                ATMBank_throughput = bankQoSTesting.getThroughput();
                System.out.println("ATMBank endToEndDelay: " + ATMBank_endToEndDelay);
                                System.out.println("ATMBank throughput: " + ATMBank_throughput);
        }
        catch(Exception e)
        {
        System.out.println(e);
        }
        }

        public void system_dynamic_test_Val()
        {

                try {
                org.omg.CosNaming.NameComponent nc = new
org.omg.CosNaming.NameComponent("ATM","");
                org.omg.CosNaming.NameComponent[] name = {nc};
                org.omg.CORBA.Object obj = root_ctx.resolve(name);
                                ATM = IATMHelper.narrow(obj);
                } catch (org.omg.CosNaming.NamingContextPackage.NotFound nf) {
                                // this is ok;
                } catch (Exception e) {
                                System.out.println("Exception resolving name/binding new context");
                                System.out.println(e.toString());
                                System.exit(1);
                }
        BankQoSTesting bankQoSTesting = new BankQoSTesting();
        SystemQoS systemQoS = null;

                try
        {
        for (int i = 0; i < 50; i++)
        {
                                bankQoSTesting.startTimer("validate");
                                        boolean val = ATM.validate("11111", "testpassword");
                                        System.out.println("Validation of Acc # 11111  & testpassword : " + val);
                                bankQoSTesting.stopTimer("validate");
```

```
		bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("validate"));
		}

		systemQoS = new SystemQoS("ATMVal");

		//these are correct codes
		systemQoS.addSystemQoS("ATMVal", "endToEndDelay", "" + bankQoSTesting.getEndToEndDelay());
		systemQoS.addSystemQoS("ATMVal", "throughput", "" + bankQoSTesting.getThroughput());

		//these are temporary codes which adjust the number for demo purpose only

		//int throughput = bankQoSTesting.getThroughput() + 1000; //operations/second
		//double endToEndDelay = (1000000/throughput); //usecond

		//systemQoS.addSystemQoS("ATMVal", "endToEndDelay", "" + endToEndDelay);
		//systemQoS.addSystemQoS("ATMVal", "throughput", "" + throughput);

				ATMVal_endToEndDelay = bankQoSTesting.getEndToEndDelay();
				ATMVal_throughput = bankQoSTesting.getThroughput();
				System.out.println("ATMVal endToEndDelay: " + ATMVal_endToEndDelay);
				System.out.println("ATMVal throughput: " + ATMVal_throughput);
		}
		catch(Exception e)
		{
		System.out.println(e);
		}
   }
}
```

SystemATMGlueTest.java

```
import java.util.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class SystemATMGlueTest{

		public ORB orb;
		public org.omg.CosNaming.NamingContext root_ctx;
		public IAccountManagement Glue_RS_CC_IAcc;
		public IValidation Glue_RS_CC_IVal;

		public double ATMGlueBank_endToEndDelay = 0.0;
		public int ATMGlueBank_throughput = 0;
		public double ATMGlueVal_endToEndDelay = 0.0;
		public int ATMGlueVal_throughput = 0;

		public SystemATMGlueTest(ORB orb, org.omg.CosNaming.NamingContext root_ctx)
		{
			this.orb = orb;
			this.root_ctx = root_ctx;
		}


		public void system_dynamic_test_Bank()
		{
			try {
			org.omg.CosNaming.NameComponent nc = new
org.omg.CosNaming.NameComponent("Glue_RS_CC_IAcc","");
```

```
                    org.omg.CosNaming.NameComponent[] name = {nc};
                    org.omg.CORBA.Object obj = root_ctx.resolve(name);
                            Glue_RS_CC_IAcc = IAccountManagementHelper.narrow(obj);
                    } catch (org.omg.CosNaming.NamingContextPackage.NotFound nf) {
                            // this is ok;
                    } catch (Exception e) {
                            System.out.println("Exception resolving name/binding new context");
                            System.out.println(e.toString());
                            System.exit(1);
                    }
                    try{
                            ICustomerManagement CT =
(ICustomerManagement)Naming.lookup("//134.68.140.101:9000/CashierTerminal");
                    BankQoSTesting bankQoSTesting = new BankQoSTesting();
                    SystemQoS systemQoS = null;

                            for (int i = 0; i < 50; i++)
            {
                          AccountInfo accountInfo_o1 = CT.openAccount("test1", "11111", 1);
                System.out.println("Account Opened Successfully\nAccount number: " +
accountInfo_o1.accountNumber +
                                  "\nAccount type: " + ((accountInfo_o1.accountType == 1) ? "Saving Account" :
"Checking Account"));

                            AccountInfo accountInfo_o0 = CT.openAccount("test2", "00000", 1);
                System.out.println("Account Opened Successfully\nAccount number: " +
accountInfo_o0.accountNumber +
                                  "\nAccount type: " + ((accountInfo_o0.accountType == 1) ? "Saving Account" :
"Checking Account"));

                            bankQoSTesting.startTimer("depositMoney");
                            Glue_RS_CC_IAcc.depositMoney(100, "11111", 1);
                                    System.out.println("deposit 11111");
                            bankQoSTesting.stopTimer("depositMoney");

            bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

                            bankQoSTesting.startTimer("depositMoney");
                            Glue_RS_CC_IAcc.depositMoney(100, "00000", 1);
                                    System.out.println("deposit 00000");
                            bankQoSTesting.stopTimer("depositMoney");

            bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

                            bankQoSTesting.startTimer("withdrawMoney");
                            Glue_RS_CC_IAcc.withdrawMoney(50, "11111", 1);
                                    System.out.println("transferring: withdraw 11111");
                            bankQoSTesting.stopTimer("withdrawMoney");

            bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

                            bankQoSTesting.startTimer("depositMoney");
                            Glue_RS_CC_IAcc.depositMoney(50, "00000", 1);
                                    System.out.print(" to deposit 00000");
                            bankQoSTesting.stopTimer("depositMoney");

            bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

                            bankQoSTesting.startTimer("withdrawMoney");
                            Glue_RS_CC_IAcc.withdrawMoney(25, "00000", 1);
                                    System.out.println("transferring: withdraw 00000");
                            bankQoSTesting.stopTimer("withdrawMoney");
```

```
	bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

				bankQoSTesting.startTimer("depositMoney");
				Glue_RS_CC_IAcc.depositMoney(25, "11111", 1);
					System.out.print(" to deposit 11111");
				bankQoSTesting.stopTimer("depositMoney");

	bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("depositMoney"));

				bankQoSTesting.startTimer("withdrawMoney");
				Glue_RS_CC_IAcc.withdrawMoney(75, "11111", 1);
					System.out.println("withdraw 11111");
				bankQoSTesting.stopTimer("withdrawMoney");

	bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

				bankQoSTesting.startTimer("withdrawMoney");
				Glue_RS_CC_IAcc.withdrawMoney(125, "00000", 1);
					System.out.println("withdraw 00000");
				bankQoSTesting.stopTimer("withdrawMoney");

	bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("withdrawMoney"));

				AccountInfo accountInfo_c1 = CT.closeAccount("11111", 1);
					System.out.println("Account Closed Successfully\nAccount number: " +
accountInfo_c1.accountNumber +
					"\nAccount type: " + ((accountInfo_c1.accountType == 1) ? "Saving Account" :
"Checking Account"));

				AccountInfo accountInfo_c0 = CT.closeAccount("00000", 1);
					System.out.println("Account Closed Successfully\nAccount number: " +
accountInfo_c0.accountNumber +
					"\nAccount type: " + ((accountInfo_c0.accountType == 1) ? "Saving Account" :
"Checking Account"));
	}

	systemQoS = new SystemQoS("ATMGlueBank");

	//these are correct codes
	systemQoS.addSystemQoS("ATMGlueBank", "endToEndDelay", "" +
bankQoSTesting.getEndToEndDelay());
	systemQoS.addSystemQoS("ATMGlueBank", "throughput", "" + bankQoSTesting.getThroughput());

	//these are temporary codes which adjust the number for demo purpose only

	//int throughput = bankQoSTesting.getThroughput() + 1000; //operations/second
	//double endToEndDelay = (1000000/throughput); //usecond

	//systemQoS.addSystemQoS("ATMGlueBank", "endToEndDelay", "" + endToEndDelay);
	//systemQoS.addSystemQoS("ATMGlueBank", "throughput", "" + throughput);

				ATMGlueBank_endToEndDelay = bankQoSTesting.getEndToEndDelay();
				ATMGlueBank_throughput = bankQoSTesting.getThroughput();
			System.out.println("ATMGlueBank endToEndDelay: " + ATMGlueBank_endToEndDelay);
				System.out.println("ATMGlueBank throughput: " + ATMGlueBank_throughput);
	}
	catch(Exception e)
	{
	System.out.println(e);
	}
	}
```

```
            public void system_dynamic_test_Val()
            {
                        try {
                        org.omg.CosNaming.NameComponent nc = new
org.omg.CosNaming.NameComponent("Glue_RS_CC_IVal","");
                        org.omg.CosNaming.NameComponent[] name = {nc};
                        org.omg.CORBA.Object obj = root_ctx.resolve(name);
                                    Glue_RS_CC_IVal = IValidationHelper.narrow(obj);
                        } catch (org.omg.CosNaming.NamingContextPackage.NotFound nf) {
                                    // this is ok;
                        } catch (Exception e) {
                                    System.out.println("Exception resolving name/binding new context");
                                    System.out.println(e.toString());
                                    System.exit(1);
                        }

            BankQoSTesting bankQoSTesting = new BankQoSTesting();
            SystemQoS systemQoS = null;

                        try
            {
            for (int i = 0; i < 50; i++)
            {
                                    bankQoSTesting.startTimer("validate");
                                                boolean val = Glue_RS_CC_IVal.validate("11111", "testpassword");
                                                System.out.println("Validation of Acc #11111 & testpassword : " + val);
                                    bankQoSTesting.stopTimer("validate");

            bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("validate"));
            }

            systemQoS = new SystemQoS("ATMGlueVal");

            //these are correct codes
            systemQoS.addSystemQoS("ATMGlueVal", "endToEndDelay", "" +
bankQoSTesting.getEndToEndDelay());
            systemQoS.addSystemQoS("ATMGlueVal", "throughput", "" + bankQoSTesting.getThroughput());

            //these are temporary codes which adjust the number for demo purpose only

            //int throughput = bankQoSTesting.getThroughput() + 1000; //operations/second
            //double endToEndDelay = (1000000/throughput); //usecond

            //systemQoS.addSystemQoS("ATMGlueVal", "endToEndDelay", "" + endToEndDelay);
            //systemQoS.addSystemQoS("ATMGlueVal", "throughput", "" + throughput);

                                    ATMGlueVal_endToEndDelay = bankQoSTesting.getEndToEndDelay();
                                    ATMGlueVal_throughput = bankQoSTesting.getThroughput();
                                    System.out.println("ATMGlueVal endToEndDelay: " + ATMGlueVal_endToEndDelay);
                                    System.out.println("ATMGlueVal throughput: " + ATMGlueVal_throughput);
            }
            catch(Exception e)
            {
            System.out.println(e);
            }
    }
}
```

Helper classes

BankQoSTesting.java

```
import java.util.*;

/**
 * This class helps the dynamic QoS testing for the banking domain example.
 * Correct sequence of using this class: startTimer(functionName), stopTimer(functionName),
getEndToEndDelay(functionName),
 * accumulateCallDelay(delay), getEndToEndDelay()/getThroughput().
 */
public class BankQoSTesting
{
   private Hashtable startingTimeTable; //keys: function name; values: Time objects
   private Hashtable stoppingTimeTable; //keys: function name; values: Time objects
   private long accumulatedDelay;
   private int accumulatedCalls; //number of accumulated calls
            private long startTime;
            private long endTime;

   /**
    * Constructor.
    */
   public BankQoSTesting()
   {
      reset();
   }

   /**
    * Reset/initialize the private members.
    */
   public void reset()
   {
      startingTimeTable = new Hashtable();
      stoppingTimeTable = new Hashtable();
      accumulatedDelay = 0;
      accumulatedCalls = 0;
   }

   /**
    * This method records the starting time.
    */
   public void startTimer(String functionName)
   {
      if(functionName != null && !functionName.trim().equals(""))
      {
                  startTime = System.currentTimeMillis();
                  Time startingTime = new Time();
                  startingTime.setTime(startTime);
         startingTimeTable.put(functionName, startingTime);
      }
   }

   /**
    * This method records the stopping time.
    */
   public void stopTimer(String functionName)
   {
      if(functionName != null && !functionName.trim().equals(""))
    {
      endTime = System.currentTimeMillis();
                    Time endingTime = new Time();
                    endingTime.setTime(endTime);
      stoppingTimeTable.put(functionName, endingTime);

    }
  }

  /**
   * This method returns end to end delay in usecond.
   */
  public long getEndToEndDelay(String functionName)
  {
    if(functionName != null && !functionName.trim().equals(""))
    {
      Time startingTime = (Time)startingTimeTable.get(functionName);
      Time endingTime = (Time)stoppingTimeTable.get(functionName);

        long delay = endingTime.getTime() - startingTime.getTime();

        startingTimeTable.remove(functionName);
        stoppingTimeTable.remove(functionName);

        return delay;
    }
    else
      return -1;
  }

  /**
   * This method accumulates delay.
   */
  public void accumulateCallDelay(long delay)
  {
    accumulatedDelay += delay;
    accumulatedCalls++;
  }

  /**
   * This method gets end to end delay (usecond).
   */
  public double getEndToEndDelay()
  {
    //sec/call
    double endToEndDelay = -1;

    if(accumulatedCalls != 0)
    {
      endToEndDelay = (accumulatedDelay + 0.0)/accumulatedCalls;
    }

    return endToEndDelay;
  }

  /**
   * This method gets throughput (operations/second).
   */
  public int getThroughput()
  {
    //calls/sec
    int throughput = -1;
```

```
        if(accumulatedCalls != 0)
        {
            throughput = (int)(1000/(accumulatedDelay/accumulatedCalls));
        }

        return throughput;
    }
}
```

ComponentQoS.java

```
import java.util.*;
import java.io.*;

public class ComponentQoS implements Serializable
{
    private Hashtable componentQoS;
    private String componentName;
    private String systemName;

    public ComponentQoS(String systemName, String componentName)
    {
        this.componentName = componentName;
        this.systemName = systemName;
        componentQoS = new Hashtable();
    }

    public void addFunctionQoS(FunctionQoS functionQoS)
    {
         if(componentName.equals(functionQoS.getComponentName()))
         {
             String key = functionQoS.getFunctionName();
             componentQoS.put(key, functionQoS);
         }
    }

    public FunctionQoS getFunctionQoS(String componentName, String functionName)
    {
        if(componentName.equals(this.componentName))
        {
            return (FunctionQoS)componentQoS.get(functionName);
        }
        else
        {
            return null;
        }
    }

    public Hashtable getComponentQoS()
    {
        return (Hashtable)componentQoS.clone();
    }

    public String getSystemName()
    {
        return systemName;
    }

    public String getComponentName()
    {
        return componentName;
    }
}
```

FunctionQoS.java

```
import java.util.*;
import java.io.*;

public class FunctionQoS implements Serializable
{
    private String componentName;
    private String functionName;
    private Hashtable functionQoSTable;

    public FunctionQoS(String componentName, String functionName)
    {
        this.componentName = componentName;
        this.functionName = functionName;
        functionQoSTable = new Hashtable();
    }

    public void addFunctionQoS(String componentName, String functionName, String QoSParameter, String value)
    {
        if(this.componentName.equals(componentName) && this.functionName.equals(functionName))
        {
            functionQoSTable.put(QoSParameter, value);
            System.out.println("In functionQoS.." + QoSParameter + value);
        }
    }

    public String getFunctionQoS(String componentName, String functionName, String QoSParameter)
    {
        if(componentName.equals(this.componentName) && functionName.equals(this.functionName))
        {
            return (String)functionQoSTable.get(QoSParameter);
        }
        else
        {
            return null;
        }
    }

    public Hashtable getFunctionQoS()
    {
        return (Hashtable)functionQoSTable.clone();
    }

    public String getComponentName()
    {
        return componentName;
    }

    public String getFunctionName()
    {
        return functionName;
    }
}
```

Time.java

```
/**
 * This class provides the implementation to get time in millisecond (msecond).
 *
 * @author Kalpana Tummala
 * @date March 2004
 * @version 1.0
 */
public class Time
{
		private long time;

		/**
	 * Constructor.
	 */
		public Time(){}

		public void setTime(long time)
		{
				this.time = time;
		}

		public long getTime()
		{
				return time;
		}
}
```

.policy

```
grant {
		permission java.net.SocketPermission "*:1024-65535", "connect,accept";
		permission java.net.SocketPermission "*:80", "connect";
		permission java.util.PropertyPermission "java.rmi.server.codebase", "read,write";
		permission java.util.PropertyPermission "com.twoab.orb2.java.orb.properties", "read";
};
grant
{
		permission java.security.AllPermission "","";
};
```

BankingException.java

```
//
// Exception definition : BankingException
//
// @author orb2 Compiler
//
public final class BankingException extends org.omg.CORBA.UserException {
  //
  // Exception member reason
  //
  public String reason;

  //
  // Default constructor
  //
  public BankingException() {
    super( BankingExceptionHelper.id() );
  }

  //
  // Constructor with fields initialization
  // @param          reason    reason exception member
  //
  public BankingException( String reason ) {
    super( BankingExceptionHelper.id() );
    this.reason = reason;
  }

  //
  // Full constructor with fields initialization
  // @param          reason    reason exception member
  //
  public BankingException( String orb_reason, String reason ) {
    super( BankingExceptionHelper.id() +" " +  orb_reason );
    this.reason = reason;
  }

}
```

BankingExceptionHelper.java

```
//
// Helper class for : BankingException
//
// @author orb2 Compiler
//
public class BankingExceptionHelper {
  private static java.lang.Object [] _extractMethods;
  static {
    try {
      Class clz = Class.forName("org.openorb.CORBA.Any");
      java.lang.reflect.Method meth = clz.getMethod("extract_Streamable", null);
      _extractMethods = new java.lang.Object[] { clz, meth };
    }
    catch(Exception ex) {
    }
    if(_extractMethods == null)
      _extractMethods = new java.lang.Object[0];
  }

  private static java.lang.reflect.Method getExtract(Class clz) {
    int len = _extractMethods.length;
    if (len == 0) return null;
    for(int i = 0; i < len; i += 2)
      if(clz.equals(_extractMethods[i]))
        return (java.lang.reflect.Method)_extractMethods[i+1];
    // unknown class, look for method.
    synchronized(org.omg.CORBA.Any.class) {
      for(int i = len; i < _extractMethods.length; i += 2)
        if(clz.equals(_extractMethods[i]))
          return (java.lang.reflect.Method)_extractMethods[i+1];
      java.lang.Object [] tmp = new java.lang.Object[_extractMethods.length+2];
      System.arraycopy(_extractMethods, 0, tmp, 0, _extractMethods.length);
      tmp[_extractMethods.length] = clz;
      try {
        tmp[_extractMethods.length+1] = clz.getMethod("extract_Streamable", null);
      }
      catch(Exception ex) {}
      _extractMethods = tmp;
      return (java.lang.reflect.Method)_extractMethods[_extractMethods.length-1];
    }
  }
```

```
//
// Insert BankingException into an any
// @param         a an any
// @param         t BankingException value
//
public static void insert( org.omg.CORBA.Any a, BankingException t ) {
  a.insert_Streamable( new BankingExceptionHolder( t ));
}

//
// Extract BankingException from an any
// @param         a an any
// @return the extracted BankingException value
//
public static BankingException extract(org.omg.CORBA.Any a) {
  if (a.type().equal(type()) == false)
    throw new org.omg.CORBA.MARSHAL();
    // streamable extraction. The jdk stubs incorrectly define the Any stub
  java.lang.reflect.Method meth = getExtract(a.getClass());
  if(meth != null) {
    try {
      org.omg.CORBA.portable.Streamable s
        = (org.omg.CORBA.portable.Streamable)meth.invoke(a, null);
      if(s instanceof BankingExceptionHolder)
        return ((BankingExceptionHolder)s).value;
    }
    catch ( IllegalAccessException ex ) {
      throw new org.omg.CORBA.INTERNAL(ex.toString());
    }
    catch ( IllegalArgumentException ex ) {
      throw new org.omg.CORBA.INTERNAL(ex.toString());
    }
    catch ( java.lang.reflect.InvocationTargetException ex ) {
      Throwable rex = ex.getTargetException();
      if(rex instanceof org.omg.CORBA.BAD_INV_ORDER)
        ; // do nothing
      else if(rex instanceof Error)
        throw (Error)rex;
      else if(rex instanceof RuntimeException)
        throw (RuntimeException)rex;
      throw new org.omg.CORBA.INTERNAL(rex.toString());
    }
    BankingExceptionHolder h = new BankingExceptionHolder(read(a.create_input_stream()));
    a.insert_Streamable(h);
    return h.value;
  }
  return read(a.create_input_stream());
}

//
// Internal TypeCode value
//
private static org.omg.CORBA.TypeCode _tc = null;
private static boolean _working = false;

//
// Return the BankingException TypeCode
// @return a TypeCode
//
public static org.omg.CORBA.TypeCode type() {
  if ( _tc == null ) {
    synchronized(org.omg.CORBA.TypeCode.class) {
      if ( _tc != null )
        return _tc;
      if ( _working )
        return org.omg.CORBA.ORB.init().create_recursive_tc( id() );
      _working = true;
      org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
      org.omg.CORBA.StructMember []_members = new org.omg.CORBA.StructMember[1];

      _members[0] = new org.omg.CORBA.StructMember();
      _members[0].name = "reason";
      _members[0].type = orb.get_primitive_tc( org.omg.CORBA.TCKind.tk_string );
      _tc = orb.create_exception_tc(id(),"BankingException",_members);
      _working = false;
    }
  }
  return _tc;
 }

 //
 // Return the BankingException IDL ID
 // @return an ID
 //
 public static String id() {
  return _id;
 }

 private final static String _id = "IDL:BankingException:1.0";

 //
 // Read BankingException from a marshalled stream
 // @param          istream the input stream
 // @return the readed BankingException value
 //
 public static BankingException read( org.omg.CORBA.portable.InputStream istream ) {
  BankingException new_one = new BankingException();

  if ( !istream.read_string().equals( id() ) )
        throw new org.omg.CORBA.MARSHAL();
  new_one.reason = istream.read_string();

  return new_one;
 }

 //
 // Write BankingException into a marshalled stream
 // @param          ostream the output stream
 // @param          value BankingException value
 //
 public static void write( org.omg.CORBA.portable.OutputStream ostream, BankingException value ) {
  ostream.write_string( id() );
  ostream.write_string(value.reason);
 }
}
```

BankingExceptionHolder.java

```
//
// Holder class for : BankingException
//
// @author orb2 Compiler
//
final public class BankingExceptionHolder
    implements org.omg.CORBA.portable.Streamable {
```

```
  //
  // Internal BankingException value
  //
  public BankingException value;

  //
  // Default constructor
  //
  public BankingExceptionHolder()
  { }

  //
  // Constructor with value initialisation
  // @param          initial      the initial value
  //
  public BankingExceptionHolder( BankingException initial ) {
    value = initial;
  }

  //
  // Read BankingException from a marshalled stream
  // @param          istream the input stream
  //
  public void _read( org.omg.CORBA.portable.InputStream istream ) {
    value = BankingExceptionHelper.read(istream);
  }

  //
  // Write BankingException into a marshalled stream
  // @param          ostream the output stream
  //
  public void _write( org.omg.CORBA.portable.OutputStream ostream ) {
    BankingExceptionHelper.write(ostream,value);
  }

  //
  // Return the BankingException TypeCode
  // @return a TypeCode
  //
  public org.omg.CORBA.TypeCode _type() {
    return BankingExceptionHelper.type();
  }
}
```

Templates

GlueCode_CS_RC_Template.txt

```
/** Glue Template for the RMI to CORBA call
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.util.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class <Glue_CS_RC> extends UnicastRemoteObject implements <CSInterface_name> {

        private String Gname;
        private static <CSInterface_name> CS_Ref;
        private static ORB orb;
        private static org.omg.CosNaming.NamingContext root_ctx ;
        private static BufferedReader inputstream;

        //constructor
        public <Glue_CS_RC>(String s) throws RemoteException {
                super();
                Gname = s;
        }

        public static void  main(String args[]) {
                System.setSecurityManager(new RMISecurityManager());
                Properties props = new Properties();
                props.setProperty("org.omg.CORBA.ORBClass", "com.twoab.orb2.core.ORBImpl");
                props.setProperty("org.omg.CORBA.ORBSingletonClass",
"com.twoab.orb2.core.ORBSingleton");

                try {
                        String init_resp_pair = "<init_resp_pair>";
                    System.out.println("************************************");
                        System.out.println("********This is GLUE for calls from RMI to CORBA for
component interaction: ");
                        System.out.println(init_resp_pair);

                        // Registering Glue component in RMI registry
                        String Gurl = "//134.68.140.101:9000/<Glue_CS_RC>";
                        <Glue_CS_RC> GCSRC = new <Glue_CS_RC>(Gurl);
                        Naming.rebind(Gurl, GCSRC);
                } catch (Exception e) {
                        System.out.println("Exception: " + e.getMessage());
                        e.printStackTrace();
                }

                // initialize ORB and locate object
                try {
                        // initialize ORB
                        orb = ORB.init(args, null);
                        // Initialize object reference for Naming Service
                        root_ctx =
org.omg.CosNaming.NamingContextHelper.narrow(orb.resolve_initial_references("NameService"));
                } catch (Exception e) {
                        System.err.println ("Exception initializing ORB:" + e);
                        return;
                }
                System.out.println("GLUE ready waiting to be configured!");
       System.out.println("*******************************************");
        }

        // Method to configure RMI component to Glue component and Glue component to CORBA component
        public void configure(String id, String kind) throws java.rmi.RemoteException {
                try{
                        // obtaining the reference of CORBA component from Naming Service
                        org.omg.CosNaming.NameComponent nc = new
org.omg.CosNaming.NameComponent(id, kind);
```

```
                    org.omg.CosNaming.NameComponent[] name = {nc};
                    org.omg.CORBA.Object obj = root_ctx.resolve(name);
                    CS_Ref = <CSInterface_name>Helper.narrow(obj);
            System.out.println("Configured GLUE using CORBA Server Loc.");
                    System.out.println("GLUE ready to send requests from RMI to CORBA");
        System.out.println("*********************************");
            }catch(Exception e){
                    System.err.println("Exception: " + e.getMessage());
                    System.err.println("-" + e);
                    e.printStackTrace();
            }
    }
// Operations implemented by CORBA component required by RMI component
<METHODS>}
```

GlueCodeMethods_CS_RC_Template.txt

```
// operation <method_name> implementation
        public <return_type> <method_name>(<parameters_signature>) throws java.rmi.RemoteException,
BankingException {
                System.out.println("GLUE received request to <method_name>");
                <return_type> value = <value>;
                double endToEndDelay = 0.0;
                int throughput = 0;
                BankQoSTesting bankQoSTesting = new BankQoSTesting();
                try{
                        bankQoSTesting.startTimer("<method_name>");
                        // call made onto CORBA component using reference
                        value = CS_Ref.<method_name>(<parameters_name>);
                        bankQoSTesting.stopTimer("<method_name>");
                bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("<method_name>"));
                } catch(BankingException e){
                        System.out.println("Glue Exception in method <method_name>: " + e);
                        e.printStackTrace();
                        throw new BankingException(e.toString());
                } catch(Exception e){
                        System.out.println("Glue Exception in method <method_name>: " + e);
                        e.printStackTrace();
                        throw new BankingException(e.toString());
                }
                System.out.println("GLUE transfering received result from Server to Client");
                endToEndDelay = bankQoSTesting.getEndToEndDelay();
                throughput = bankQoSTesting.getThroughput();
                System.out.println("endToEndDelay: " + endToEndDelay);
                System.out.println("throughput: " + throughput);

        System.out.println("***************************************************************");
                return value;
        }
```

GlueCodeVoidMethods_CS_RC_Template.txt

```
// operation <method_name> implementation
        public <return_type> <method_name>(<parameters_signature>) throws java.rmi.RemoteException,
BankingException {
                System.out.println("GLUE received request to <method_name>");
                double endToEndDelay = 0.0;
                int throughput = 0;
                BankQoSTesting bankQoSTesting = new BankQoSTesting();
                try{
                        bankQoSTesting.startTimer("<method_name>");
                        // call made onto CORBA component using reference
                        CS_Ref.<method_name>(<parameters_name>);
                        bankQoSTesting.stopTimer("<method_name>");
                bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("<method_name>"));
                } catch(BankingException e){
                        System.out.println("Glue Exception in method <method_name>: " + e);
                        e.printStackTrace();
                        throw new BankingException(e.toString());
                } catch(Exception e){
                        System.out.println("Glue Exception in method <method_name>: " + e);
                        e.printStackTrace();
                        throw new BankingException(e.toString());
                }
                System.out.println("GLUE transfering received result from Server to Client");
                endToEndDelay = bankQoSTesting.getEndToEndDelay();
                throughput = bankQoSTesting.getThroughput();
                System.out.println("endToEndDelay: " + endToEndDelay);
                System.out.println("throughput: " + throughput);

        System.out.println("*****************************************************************");
        }
```

GlueCode_RS_CC_Template.txt

```
/** Glue Template for the CORBA to RMI call
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import java.util.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;

public class <Glue_RS_CC> extends <RSInterface_name>POA {

    private static ORB orb;
    private static POA root_poa;
    private static org.omg.CosNaming.NamingContext root_ctx;

    static <RSInterface_name> RS_Ref;

    //constructor
    <Glue_RS_CC>(ORB orb, POA root_poa, org.omg.CosNaming.NamingContext root_ctx) {
        _this_object(orb);
        this.orb = orb;
        this.root_poa = root_poa;
        this.root_ctx = root_ctx;
    }

    public static void main(String[] args) {
        Properties props = new Properties();
        props.setProperty("org.omg.CORBA.ORBClass", "com.twoab.orb2.core.ORBImpl");
        props.setProperty("org.omg.CORBA.ORBSingletonClass",
"com.twoab.orb2.core.ORBSingleton");

        System.setSecurityManager(new RMISecurityManager());
```

```
                try {
                        String init_resp_pair = "<init_resp_pair>";

        System.out.println("*****************************************************************");
                        System.out.println("********This is GLUE for calls from CORBA to RMI for
component interaction: ");
                        System.out.println(init_resp_pair);

                        // Initialize ORB and Root POA
                        orb = ORB.init(args, null);
                        org.omg.CORBA.Object obj = orb.resolve_initial_references("RootPOA");
                        root_poa = POAHelper.narrow(obj);
                        // Initialize object reference for Naming Service
                        root_ctx = org.omg.CosNaming.NamingContextHelper.narrow
(orb.resolve_initial_references("NameService"));
                        // Create and activate servant/reference
                        <Glue_RS_CC> GRSCC = new <Glue_RS_CC>(orb, root_poa, root_ctx);
                        <RSInterface_name> GRSCC_Ref = GRSCC._this();
                        root_poa.the_POAManager().activate();
                        // Initialize Glue object reference in Naming Service
                        org.omg.CosNaming.NameComponent nc = new
org.omg.CosNaming.NameComponent("<Glue_RS_CC>", "");
                        org.omg.CosNaming.NameComponent[] name = {nc};

                        try {
                                root_ctx.resolve(name);
                                root_ctx.bind_new_context(name);
                        } catch (org.omg.CosNaming.NamingContextPackage.NotFound nf) {
                                // this is ok;
                        } catch (org.omg.CosNaming.NamingContextPackage.AlreadyBound ab) {
                                // this is ok;
                        } catch (Exception e) {
                                System.out.println("Exception resolving name/binding new context");
                                System.out.println(e.toString());
                                System.exit(1);
                        }

                        root_ctx.rebind(name, GRSCC_Ref);
                        // Wait for requests
                        System.out.println("GLUE ready waiting to be configured!");

        System.out.println("*****************************************************************");
                        orb.run();
                } catch(Exception e) {
                        e.printStackTrace();
                        System.err.println("Exception occurred " + e.toString());
                }
        }

        // Method to configure CORBA component to Glue component and Glue component to RMI component
        public void configure(String RSLocation){
                try {
                        // obtaining the reference of RMI component from RMI Registry
                        RS_Ref = (<RSInterface_name>) Naming.lookup(RSLocation);
                System.out.println("Configured GLUE using RMI Server Loc.");
                        System.out.println("GLUE ready to send requests from CORBA to RMI");

        System.out.println("*****************************************************************");
                } catch(Exception e) {
                        System.out.println("<Glue_RS_CC> Exception: " + e.getMessage());
                        e.printStackTrace();
                }
```

```
		}

// Operations implemented by RMI component required by CORBA component
<METHODS>}
```

GlueCodeMethods_RS_CC_Template.txt

```
// operation <method_name> implementation
	public <return_type> <method_name>(<parameters_signature>) throws BankingException{
		System.out.println("GLUE received request to <method_name>");
		<return_type> value = <value>;
		double endToEndDelay = 0.0;
		int throughput = 0;
		BankQoSTesting bankQoSTesting = new BankQoSTesting();
		try{
			bankQoSTesting.startTimer("<method_name>");
			// call made onto RMI component using reference
			value = RS_Ref.<method_name>(<parameters_name>);
			bankQoSTesting.stopTimer("<method_name>");
		bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("<method_name>"));
		} catch(BankingException e){
			System.out.println("Glue Exception in method <method_name>: " + e.toString());
			e.printStackTrace();
			throw new BankingException(e.toString());
		} catch(Exception e){
			System.out.println("Glue Exception in method <method_name>: " + e);
			e.printStackTrace();
			throw new BankingException(e.toString());
		}
		System.out.println("GLUE transfering received result from Server to Client");
		endToEndDelay = bankQoSTesting.getEndToEndDelay();
		throughput = bankQoSTesting.getThroughput();
		System.out.println("endToEndDelay: " + endToEndDelay);
		System.out.println("throughput: " + throughput);

	System.out.println("***************************************************************");
		return value;
	}
```

GlueCodeVoidMethods_RS_CC_Template.txt

```
// operation <method_name> implementation
	public <return_type> <method_name>(<parameters_signature>) throws BankingException{
		System.out.println("GLUE received request to <method_name>");
		double endToEndDelay = 0.0;
		int throughput = 0;
		BankQoSTesting bankQoSTesting = new BankQoSTesting();
		try{
			bankQoSTesting.startTimer("<method_name>");
			// call made onto RMI component using reference
			RS_Ref.<method_name>(<parameters_name>);
			bankQoSTesting.stopTimer("<method_name>");
		bankQoSTesting.accumulateCallDelay(bankQoSTesting.getEndToEndDelay("<method_name>"));
		} catch(BankingException e){
			System.out.println("Glue Exception in method <method_name>: " + e.toString());
			e.printStackTrace();
			throw new BankingException(e.toString());
		} catch(Exception e){
			e.printStackTrace();
			throw new BankingException(e.toString());
		}
		System.out.println("GLUE transfering received result from Server to Client");
```

```
                endToEndDelay = bankQoSTesting.getEndToEndDelay();
                throughput = bankQoSTesting.getThroughput();
                System.out.println("endToEndDelay: " + endToEndDelay);
                System.out.println("throughput: " + throughput);

        System.out.println("****************************************************************");
        }
```

GlueConfigure_CS_RC_Template.txt

```
/** Glue Configure Template for the RMI to CORBA call
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.util.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import org.omg.CORBA.*;

public class <Configure_CS_RC> {

        private String RCLocation = "<RCLocation>";
        private String GLUELocation = "<GLUELocation>";
        private String CSid = "<CSid>";
        private String CSkind = "<CSkind>";
        private String CScomponentName = "<CSComponentName>";

        // constructor
        public <Configure_CS_RC>() {

        }

        // Method to configure RMI component to Glue component and Glue component to CORBA component
        public void configureComponents() {
                try {
                        // obtaining the reference of RMI component
                        IConfiguration client = (IConfiguration)Naming.lookup(RCLocation);

                        StringTokenizer st = new StringTokenizer(GLUELocation, "/");
                        String GLocation = st.nextToken();
                        GLocation = GLocation + "/" + st.nextToken();
                        System.out.println(GLocation);

                        // configuring RMI component to Glue component
                        client.configure(GLocation, CScomponentName);

                        System.out.println("Configured RMI client to location of GLUE");

        System.out.println("****************************************************************");
                        // obtaining the reference of Glue component
                        <GLUEInterface> adapter = (<GLUEInterface>)Naming.lookup(GLUELocation);
                        // configuring Glue component to CORBA component
                        adapter.configure(CSid, CSkind);
                        System.out.println("Configured GLUE to location of CORBA");

        System.out.println("****************************************************************");
                } catch(Exception e) {
                        System.out.println("Glue Configure lookup Exception: " + e);
                        try{
                                int g = System.in.read();
                        }catch(Exception io){
                                io.printStackTrace();
                        }
                                e.printStackTrace();

                }
        }

        public static void main(String args[]) {
//              System.setSecurityManager(new RMISecurityManager());
                <Configure_CS_RC> glueCon = new <Configure_CS_RC>();

                String init_resp_pair = "<init_resp_pair>";

        System.out.println("**************************************************************");
                        System.out.println("*******This is Glue to configure components: ");
                System.out.println(init_resp_pair);

                System.out.println("Press ENTER to configure");
                try{
                        int x = System.in.read();
                        glueCon.configureComponents();
                        int y = System.in.read();
                }catch(Exception io){
                        io.printStackTrace();
                }
        }
}
```

GlueConfigure_RS_CC_Template.txt

```
/** Glue Configure Template for the CORBA to RMI call
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import org.omg.CORBA.*;
import java.util.*;
import java.io.*;

public class <Configure_RS_CC> {

        private String CCid = "<CCid>";
        private String CCkind = "<CCkind>";
        private String GLUEid = "<GLUEid>";
        private String GLUEkind = "<GLUEkind>";
        private String RSLocation = "<RSLocation>";
        private String RScomponentName = "<RSComponentName>";
        private static ORB orb;
        private static org.omg.CosNaming.NamingContext root_ctx;
        private static <GLUEInterface_name> Glue;
        private static <CCInterface_name> client;

        // constructor
        public <Configure_RS_CC>() {
		}

		// Method to configure CORBA component to Glue component and Glue component to RMI component
		public void configureComponents() {
			try {
				// obtaining the reference of CORBA component
				org.omg.CosNaming.NameComponent nc_init = new
org.omg.CosNaming.NameComponent(CCid, CCkind);
				org.omg.CosNaming.NameComponent[] name_init = {nc_init};
				org.omg.CORBA.Object obj_init = root_ctx.resolve(name_init);
				client = <CCInterface_name>Helper.narrow(obj_init);
				// configuring CORBA component to Glue component
				client.configure(GLUEid, RScomponentName);


		System.out.println("*****************************************************************");
			System.out.println("Configured CORBACLient using GLUE Loc.");


		System.out.println("*****************************************************************");
				// obtaining the reference of Glue component
				org.omg.CosNaming.NameComponent nc_adap = new
org.omg.CosNaming.NameComponent(GLUEid, GLUEkind);
				org.omg.CosNaming.NameComponent[] name_adap = {nc_adap};
				org.omg.CORBA.Object obj_adap = root_ctx.resolve(name_adap);
				Glue = <GLUEInterface_name>Helper.narrow(obj_adap);
				// configuring Glue component to RMI component
				Glue.configure(RSLocation);
			System.out.println("Configured GLUE to Location of Server.");

		System.out.println("*****************************************************************");
			} catch(Exception e) {
				System.out.println("Glue lookup Excption: " + e);
				try{
					int g = System.in.read();
				}catch(Exception io){
					io.printStackTrace();
				}
				e.printStackTrace();
			}
		}

		public static void main(String args[]) {
			try {
				String init_resp_pair = "<init_resp_pair>";

		System.out.println("*****************************************************************");
				System.out.println("*******This is Glue to configure components********");
				System.out.println(init_resp_pair);
				orb = ORB.init(args, null);
				// Initialize object reference for Naming Service
				root_ctx = org.omg.CosNaming.NamingContextHelper.narrow
(orb.resolve_initial_references("NameService"));
			} catch (Exception e) {
				System.out.println("Exception: " + e);
				try{
					int g = System.in.read();
				}catch(Exception io){
					io.printStackTrace();
				}
			}
			<Configure_RS_CC> glueCon = new <Configure_RS_CC>();
```

```
                System.out.println("Press ENTER to configure");
                try{
                        int x = System.in.read();
                        glueCon.configureComponents();
                        int y = System.in.read();
                }catch(Exception io){
                                io.printStackTrace();
                }
        }
}
```

GlueInterface_CS_RC_Template.txt

```
/** Glue Interface Template for the RMI to CORBA call
 *
 *  @author Kalpana Tummala
 *  @date March 2004
 *  @version 1.0
 */

import java.rmi.Remote;

public interface <CSInterface_name> extends java.rmi.Remote {

        public void configure(String id, String kind) throws java.rmi.RemoteException;

<METHODS>}
```

GlueInterfaceMethods_CS_RC_Template.txt

```
        public <return_type> <method_name>(<parameters_signature>) throws java.rmi.RemoteException,
BankingException;
```

GlueInterface_RS_CC_Template.txt

```
// Glue Interface Template for the CORBA to RMI call
//
// @author Kalpana Tummala
// @date March 2004
// @version 1.0

exception BankingException { string reason; };

  interface <RSInterface_name> {

    // Operation declarations:

        void configure(in string RSLocation);

<METHODS>  };
```

GlueInterfaceMethods_RS_CC_Template.txt

```
    <return_type> <method_name>(<parameters_signature>) raises (BankingException);
```

SuperBank_Spec.txt

```
System Name: SuperBank
C1-->UMM1
C1: CashierTerminal
```

```
UMM1: C:/Kalpana/SuperBank1/CT/CashierTerminal_spec.xml
C2-->UMM2
C2: CashierValidationServer
UMM2: C:/Kalpana/SuperBank1/CashVS/CashierValidationServer_spec.xml
C3-->UMM3
C3: CustomerValidationServer
UMM3: C:/Kalpana/SuperBank1/CustVS/CustomerValidationServer_spec.xml
C4-->UMM4
C4: ATM
UMM4: C:/Kalpana/SuperBank1/ATM/ATM_spec.xml
C5-->UMM5
C5: TransactionServerManager
UMM5: C:/Kalpana/SuperBank1/TSM/TransactionServerManager_spec.xml
C6-->UMM6
C6: DeluxeTransactionServer
UMM6: C:/Kalpana/SuperBank1/DTS/DeluxeTransactionServer_spec.xml
C7-->UMM7
C7: AccountDatabase
UMM7: C:/Kalpana/SuperBank1/AD/AccountDatabase_spec.xml
comm: request-reply
System-->C1commC2;C1commC5;C1commC6;C4commC6;C4commC3;C4commC5;C5commC6;C6commC7
Placement: Centralized
```
